MASTER EN SCIENCES INFORMATIQUES

Étude et implémentation de méthodes de coloration de graphes

Campers, G.

Award date:
1985

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Etude et implémentation de méthodes de coloration de graphes

G. CAMPERS

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

Je remercie sincèrement

- Monsieur O. HENKES pour l'aide précieuse qu'il m'a accordée et pour les conseils et encouragements qu'il ma prodigués durant la réalisation de ce mémoire.

- Monsieur J.P. LECLERQ pour les conditions de travail qu'il m'a procurées et Monsieur J. PAQUET pour son aide efficace et sa disponibilité à toute épreuve.

Je remercie aussi Madame S. BARTHOLOME qui a assuré avec autant de gentillesse que de compétence la dactylographie du manuscrit.

J'adresse toute ma gratitude à ma fiancée, Véronique, pour la patience et la compréhension dont elle a fait preuve durant cette année d'études.

Enfin, un grand merci à tous ceux, et ils sont nombreux, que j'ai dérangés dans l'accomplissement de ce travail.

TABLE DES MATIERES

CHAPITRE III. - LES METHODES EXACTES

CHAPITRE IV. - POINTS DELICATS DE L'IMPLEMENTATION

I N T R O D U C T I O N

La coloration de graphes est d'application dans une grande variété des problèmes complexes relevant de l'optimisation. En particulier, la résolution de "conflits" et la répartition optimale d'évènements incompatibles trouvent souvent leur solution dans la coloration de graphes. Les exemples de tels problèmes englobent : l'établissement d'un horaire d'examens dans le laps de temps le plus court, de telle manière qu'aucun étudiant ne doive participer à 2 épreuves simultanément ou, le rangement de produits chimiques, sur un nombre minimum d'étagères, de manière à stocker deux substances, qui interagissent mutuellement dangereusement, soient rangées sur des étagères distinctes.

Dans chacun des problèmes exposés ci-dessus, les contraintes sont généralement exprimables sous la forme de paires d'objets incompatibles (ex.: deux substances chimiques qui ne peuvent être stockées sur la même étagère). La structure de graphe est alors très utile pour modéliser de telles incompatibilités. En effet, chaque objet peut être représenté par un noeud et chaque incompatibilité par une arête joignant deux noeuds. Une coloration d'un tel graphe consiste alors à partitionner l'ensemble des objets en blocs (ou couleurs) de telle manière que deux objets incompatibles soient dans deux blocs distincts. Donc, une solution optimale aux problèmes décrits précédemment, peut être trouvée en déterminant les colorations minimales (i.e. utilisant un minimum de couleurs) pour les graphes correspondants.

Comme le problème de colorer un graphe est NP-complet (/15/); il n'existe pas d'algorithme qui, pour tout graphe, colore, de manière optimale, les noeuds de celui-ci en un temps borné par une expression polynominale du nombre de sommets. Puisque des algorithmes caractérisés par un "temps exponentiel" sont

prohibitivement coûteux pour une application sur des problèmes de grande taille, l'intérêt s'est porté sur le développement de méthodes heuristiques qui produisent usuellement une bonne, mais pas nécessairement optimale, coloration pour n'importe quel graphe en un temps raisonnable. Malheureusement, on peut lire dans / 1 / que ces dernières méthodes fournissent, pour certains graphes particuliers, des solutions qui s'écartent de manière significative du nombre chromatique ou nombre minimum de couleurs.

Après avoir rappelé, lors de préliminaires, quelques notions de la théorie des graphes qui nous paraissent indispensables pour la lecture de cet ouvrage, nous consacrons deux chapîtres (II et III) à l'exposé (les principes de base, l'algorithme, un exemple et une analyse de la complexité théorique) de vingt méthodes heuristiques et de cinq méthodes exactes. Parmi ces cinq dernières, se trouve l'"énumération implicite", un algorithme que nous avons mis au point nous-même.

Ces 25 méthodes ont donné lieu à une implémentation en FORTRAN 77 sur VAX/VMS dont l'explication de quelques points délicats constitue le chapitre IV. Cette implémentation nous a permis de tester les algorithmes décrits sur des graphes, générés aléatoirement, dont le nombre de sommets valait respectivement 20, 50, 75, 100. Pour chaque nombre de noeuds 5 graphes d'une densité $\mu = 0.2 ; 0.4 ; 0.6 ; 0.8$ ont été construits, où la densité μ est définie par d/n (où d est le degré moyen d'un noeud du graphe). Ainsi, 80 graphes aléatoires forment nos jeux de test. Ces derniers ont été obtenus à l'aide d'une procédure, décrite en annexe 1, qui génère des graphes aléatoires dont le nombre chromatique est connu. L'existence de cette procédure (/2/), qui a souvent fait défaut jusqu'à présent dans la littérature, fournit une méthode pour estimer la précision d'un algorithme de coloration.

Lors des tests, nous nous sommes intéressés à deux performances des méthodes de coloration : le temps CPU consommé et le nombre de couleurs utilisées.
Cette dernière mesure, grâce au fait que le nombre chromatique des graphes utilisés pour les tests, soit connu, permet d'évaluer justement la qualité d'une coloration fournie par une méthode heuristique. Les principaux résultats ainsi enregistrés, des graphiques les illustrant et les conclusions que l'on peut en tirer clôturent les chapîtres dédiés aux méthodes auxquelles ils se rapportent.

Enfin, dans le chapître V., nous tirons les conclusions de ce travail et nous discutons, au vu du résultat des tests, l'utilisation des différentes méthodes dans des applications pratiques.

*
* *

CHAPITRE I. - PRELIMINAIRES

CHAPITRE I. : Préliminaires

Dans ce chapitre, nous présenterons les différents concepts et notations utilisés dans ce travail.

Un graphe G est représenté par un couple (X , V), où X est un ensemble de noeuds ou sommets et V est un ensemble d'arêtes reliant des éléments de X .

Parmi les graphes nous nous intéresserons uniquement aux 1 - graphes sans boucle , i.e. des graphes G (X , V) tels que deux noeuds de X sont connectés par au plus une arête, et toute arête de V relie toujours deux noeuds distincts.

Dorénavant, et puisque notre étude se limite seulement à ceux-ci, nous restreindrons le concept de graphe à celui de 1 - graphe sans boucles.

Considérons, maintenant le graphe G (X, V) où $X = \{x_1, x_2, \ldots x_n\}$. on définit la matrice d'adjacence de G , A de la manière suivante :

$A_{ij} = 1$ ssi $\exists$ une arête de V qui relie x_i à x_j
$= 0$ sinon

Remarquons immédiatement que $A_{ii} = 0$ et $A_{ij} = A_{ji}$.

- On dira que x_i est adjacent à x_j ssi $\exists$ une arête de V qui relie x_i à x_j
- Le degré d_i de x_i est le nombre d'adjacents de x_i . Cette définition implique que $d_i = \sum_j A_{ij}$
- Un sommet dont le degré est nul et donc qui ne possède aucun adjacent est dit isolé
- Le graphe complément de G (X, V) est $\overline{G}$ $(X, \overline{V})$

où $\overline{V} = \{u \mid u \notin V \wedge u$ n'est pas une boucle$\}$

- Soit $S \subseteq X$; <u>le sous-graphe engendré</u> par S, noté $\langle S \rangle$, est (S, V') où $u \in V' \Leftrightarrow u \in V$ et u connecte 2 noeuds de S
- Une chaîne est une suite d'arêtes $(\bar{u}_1 , \ldots \bar{u}_q)$ telle que l'arête $\bar{u}_i$ $(1 < i < q)$ est reliée à $\bar{u}_{i-1}$ par une de ses extrémités et à $\bar{u}_{i+1}$ par l'autre. De plus, lorsqu'il y a répétition des arêtes, chaque arête est parcourue dans le même sens. Cette notion permet de définir la relation d'équivalence sur les sommets de G :

  $x_i \; R \; x_j \Leftrightarrow \quad x_i = x_j \quad V \; \exists$ une chaîne qui relie x_i et x_j .

  Cette relation d'équivalence, qu'on appellera <u>relation de connexité</u>, permet de diviser X en classe d'équivalence : <u>les composantes connexes</u> du graphe G . S'il n'existe qu'une seule composante connexe, le graphe est dit <u>connexe</u>.
- Soit $C \subseteq X$, on dit que C est une <u>clique de G (X,V)</u> ssi $\forall \; x_i , x_j \in C, \exists$ une arête de V qui joint x_i et x_j .

  Dans le cas où C = X tout entier, G (X,V) est dit <u>complet</u>.
- Un ensemble de sommets de G est <u>indépendant</u> s'il n'en existe pas deux d'entre eux qui sont adjacents.
- Un <u>ensemble indépendant</u> de sommets, M, est <u>maximal</u> ssi il n'existe pas d'ensemble indépendant contenant M dans G .
- Une <u>clique C</u> est <u>maximale</u> ssi il n'existe pas de clique contenant C dans G
- <u>Une coloration du graphe</u> G (X,V) est une fonction f partout définie de l'ensemble des noeuds X vers un ensemble de couleurs= 1, 2, 3, ... t.q. 2 sommets adjacents se voient attribuer 2 couleurs distinctes.

. Le nombre minimum de couleurs requises pour colorer G est appelé <u>nombre chromatique</u> du graphe G et noté $\chi(G)$.

Nous terminerons en attirant l'attention du lecteur sur le fait que l'on travaille toujours avec des graphes "non-orientés" (i.e. on effectue aucune distinction entre extrémité initiale et extrémité terminale d'une arête) et en signalant la correspondante suivante avec les concepts "orientés."

arête $\longleftrightarrow$ arc

chaîne $\longleftrightarrow$ chemin

connexité $\longleftrightarrow$ connexité forte

*

* *

CHAPITRE II. - LES METHODES NON-EXACTES

II.1 Méthodes de coloration séquentielle

II.2 Les méthodes de saturation

II.3 Méthodes diverses non-exactes

II.4 Résultats de tests

CHAPITRE II. : Les méthodes non-exactes

Nous donnons, ici, les principes de base, l'algorithme et la complexité théorique de plusieurs méthodes heuristiques. Celles-ci ne fournissent qu'une approximation du nombre chromatique du graphe qu'elles colorent. Cependant, elles sont, pour la plupart, peu coûteuses en temps C P U .

Nous terminerons ce chapitre en exposant les conclusions que nous avons tirées du test de ces méthodes sur quelque 80 graphes.

(II.1) METHODES DE COLORATION SEQUENTIELLE

Le caractère séquentiel des méthodes présentées dans ce paragraphe suggère une subdivision de celles-ci en deux classes, suivant qu'on examine la séquentialité du point de vue des sommets ou des couleurs..

L'idée de base des algorithmes adhérant à la première optique, est de classer les noeuds dans un ordre lié à leur difficulté de coloration et de les colorer alors un à un dans cet ordre en leur donnant la plus petite couleur possible; tandis que l'approche opposée conduit à attribuer une couleur au plus grand ensemble possible de sommets avant de considérer la couleur suivante et à recommencer ce processus jusqu'à épuisement des sommets à traiter.

(II.1.1) LES METHODES SEQUENTIELLES PAR SOMMETS

Les traitements proposés par ces méthodes peuvent se décomposer en 2 phases :

(i) une phase d'ordonnancement des noeuds.
(ii) une phase de coloration des noeuds dans l'ordre établi en (i) .

La classification, que nous établissons de ces méthodes, repose sur le critère d'ordonnancement qu'elles utilisent en (i). Sinon, l'algorithme réalisant (ii) est identique pour toutes et peut s'énoncer, si on considère que le graphe à colorer G (X,E) comporte n sommets :

Pour i de 1 à n

Soit x_i le sommet classé $i^{ième}$.

choisir c la plus petite couleur possible pour x_i

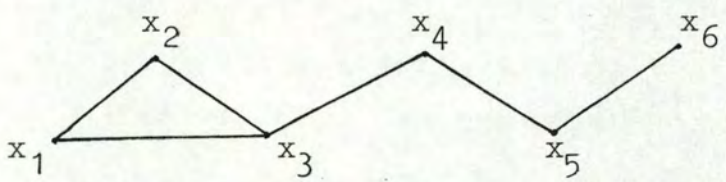
colorer x_i avec c

interdire la couleur c aux adjacents de x_i

Fin pour

Exemple :

Soit le graphe suivant à colorer



Supposons que l'ordre de coloration soit x_1, x_2 ... , x_6 :

Le raisonnement de coloration est alors :

- donner la couleur 1 à x_1 ; interdire la couleur 1 pour x_2 et x_3.
- " 2 à x_2 ; " 2 " x_1 et x_3.
- " 3 à x_3 ; " 3 " x_1,x_2,x_4.
- " 1 à x_4 ; " 1 " x_3 et x_5.
- " 2 à x_5 ; " 2 " x_4 et x_6.
- " 1 à x_6 ; " 1 " x_5

Ainsi, on obtient la coloration 1, 2, 3, 1, 2, 1, c-à-d. qu'il faut 3 couleurs pour colorer le graphe. Si on avait considéré l'ordre x_3 , x_1 , x_2 , x_4 , x_5 , x_6 , on aurait trouvé la coloration 2, 3, 1, 2, 1, 2.

Le résultat de la coloration dépend donc fortement de l'ordre dans lequel les sommets sont examinés.

Avant d'énumérer les méthodes regroupées dans cette rubrique, nous allons donner une analyse de complexité de l'algorithme décrit ci-dessus :

(i) on effectue n passages dans la boucle "Pour"
(ii) Lors de chacun de ces passages :

- La coloration d'un sommet coûte au plus $\mathcal{O}(k)$ temps de calcul
  où
  k est le nombre de couleurs utilisées par la méthode.
- La procédure d'interdiction d'une couleur aux adjacents d'un noeud donné utilise $\mathcal{O}(d)$ temps de calcul (où d est le degré moyen d'un sommet du graphe).

Donc l'algorithme examiné exige $\mathcal{O}(n.(k+d))$ temps de calcul. Puisque $k \leqslant n$ et $d \leqslant n$, on obtient finalement $\mathcal{O}(n^2)$ temps de calcul.

(II.1.1.1) ORDRE ALEATOIRE

On attribue à chaque noeud x_i un poids aléatoire p_i , ensuite on ordonne les sommets de la manière suivante :

$$x_i < x_j \iff p_i \leqslant p_j$$

Remarquons tout de suite que cet ordonnancement ne tient aucun compte de la structure du graphe à traiter.
L'analyse de complexité donne :

(i) l'attribution des poids aléatoires aux noeuds exige $\mathcal{O}(n)$ temps de calcul.

(ii) Si l'ordonnancement des noeuds suivant les poids aléatoires est exécuté grâce à un Heapsort, on sait alors (cfr / 3 /), qu'il nécessite $\mathcal{O}(n \log_2 n)$ temps calcul.

(iii) Comme nous l'avons établi ci-dessus la coloration elle-même utilise $\mathcal{O}(n^2)$ temps calcul.

Donc cette méthode exige $\mathcal{O}(n^2)$ temps de calcul.

(II.1.1.2) ORDRE LARGEST-FIRST (/1/ , /2/ , /4/ , /5/)

La considération sous-jacente à la méthode est qu'un noeud est d'autant plus difficile à colorer qu'il a beaucoup d'adjacents.
En effet, au moment de la coloration d'un tel noeud, il est possible qu'on doive tenir compte d'un nombre assez important de contraintes inhérentes au nombre d'adjacents déjà colorés. C'est pourquoi, l'idée maîtresse de la méthode est de colorer en premier lieu de tels sommets.

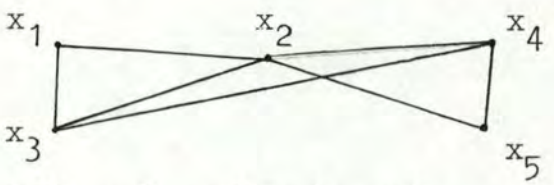
D'où l'algorithme L F :

(i) calculer le degré d_i de chaque noeud du graphe

(ii) ordonner les noeuds suivant leur degré

(iii) colorer les noeuds dans l'ordre établi.

EXEMPLE :



. Ordre : x_2 , x_3 , x_4, x_1 , x_5

L'analyse de complexité donne :

(i) Puisque $d_i = \sum_j A_{ij}$ (où A_{ij} est la matrice d'adjacence), le calcul des degrés exige n x(n additions) donc $\mathcal{O}(n^2)$ temps calcul.

(ii) l'ordonnancement des noeuds, que l'on peut réaliser par un Heapsort, se réalise en $\mathcal{O}(n \log_2 n)$ temps de calcul.

(iii) la coloration requiert $\mathcal{O}(n^2)$ temps de calcul.

Ce qui donne pour cette méthode : $\mathcal{O}(n^2)$ temps de calcul.

(II.1.1.3.) <u>ORDRE SMALLEST-LAST</u> (/1/ , /2/ , /4/)

Cette méthode est basée sur le même principe que la précédente. Toutefois, la réalisation de l'ordonnancement des noeuds est différente; les degrés sont recalculés (sur le graphe restant) chaque fois qu'un sommet a été classé (en fin de liste).

d'où l'algorithme SL (X = ensemble des n sommets du graphe)

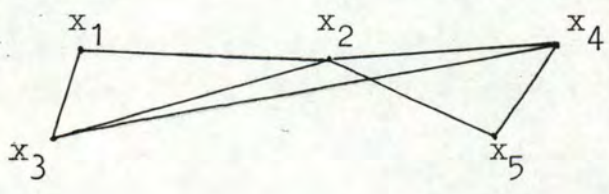
(i) Soit x_n tel que degré $(x_n) = \min_{1 \leqslant i \leqslant n}$ degré (x_i)

(ii) <u>pour</u> i = n - 1, n - 2,...,2,1

x_i est le sommet de degré minimal dans le graphe engendré par $X - \{x_n , x_{n-1},\ldots,x_{i+1}\}$

(iii) colorer les noeuds dans l'ordre établi.

<u>EXEMPLE</u> :



ordre : x_5 , x_4 , x_3 , x_2 , x_1

L'analyse de complexité donne :

Pour i quelconque, la sélection de x_i donne lieu :
(i) au calcul du degré de chaque noeud
" restant " $\longrightarrow$ i opérations

(ii) à la recherche du noeud de degré minimum
$\longrightarrow$ i opérations

L'ordonnancement exige donc $\sum_{i=1}^{n} 2i = n.(n+1)$ opérations

par conséquent $\Theta(n^2)$ temps de calcul.

De plus, puisque les opérations de coloration proprement dites utilisent $\Theta(n^2)$ temps de calcul, la méthode Smallest-last requiert $\Theta(n^2)$ temps de calcul.

(II.1.1.4.) ORDONNANCEMENT SUIVANT LE DEGRE GENERALISE (/1/ , /6/)

Si on applique la méthode Largest-first, le sommet coloré à n'importe quelle étape du processus de coloration est le noeud "colorable" de degré maximum. Cependant, des complications apparaissent quand 2 sommets ou plus ont le même degré. On peut montrer (/1/) qu'une erreur dans l'ordre de sélection des noeuds peut provoquer un accroissement du nombre de couleurs utilisées.

Considérons $A = (A_{ij})$ la matrice d'adjacence du graphe et d le vecteur dont la composante d_i vaut le degré du $i^{ième}$ sommet au graphe. Nous avons vu dans les préliminaires que

$$d_i = \sum_j A_{ij}$$

C'est ce vecteur qui est utilisé dans la procédure Largest-first pour décider quel sera le noeud suivant à colorer.

La théorie des matrices nous apprend que ce vecteur d n'est rien d'autre qu'une approximation du vecteur propre dominant de la matrice A. En effet, une meilleure approximation d^2 peut être obtenue grâce à une multiplication matricielle et en général, le processus itératif

$$d_i^{P+1} = \sum_j A_{ij} \, d_j^p \qquad (I)$$

converge vers le vecteur propre dominant de A quand $p \to \infty$

Si l'on modifie la méthode Largest-first en utilisant d^p à la place de d , un amélioration significative de la méthode peut être remarquée. En outre, il a été montré que l'itération ci-dessus ne doit pas être exécutée un grand nombre de fois, en effet, si le graphe possède n sommets alors

$$p = \left| \sqrt[3]{n} \right|$$

est généralement suffisant. De plus, d_i^p est appelé <u>degré généralisé</u> du $i^{ième}$ noeud du graphe

D'où l'algorithme DGEN

(i) Calcul du degré généralisé des noeuds du graphe
(ii) ordonner les noeuds suivant leur degré généralisé
(iii) colorer les noeuds dans l'ordre établi.

L'analyse de complexité donne :

(i) Le calcul du degré généralisé implique (p - 1) exécutions de l'itération (I) (avec $p = \left| \sqrt[3]{n} \right|$). L'itération (I) est une multiplication d'une matrice (n x n) et d'un vecteur (n x 1), et de ce fait, exige n x (n multiplications + (n - 1) additions). Donc le calcul du degré généralisé coûte $\delta(n^2)$ temps de calcul.

(ii) l'ordonnancement des noeuds suivant le degré généralisé est réalisé grâce à un HEAPSORT, et donc nécessite $\Theta(n.\log_2 n)$ temps de calcul.

(iii) les opérations de coloration prennent $\Theta(n^2)$ temps de calcul.

Donc au total la méthode du degré généralisé requiert $\Theta(n^2)$ temps de calcul.

(II.1.1.5.) ORDRE "DEPTH-FIRST SEARCH" (/1/)

L'idée de base de la méthode est de colorer les noeuds dans l'ordre où ils se présentent dans une depth-first search.

Une depth-first search est une méthode de parcours d'un graphe.

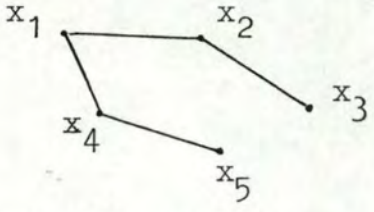
Elle procède de la manière suivante :

(i) Initialement : tous les sommets sont inexplorés.

(ii) On choisit un sommet de départ

(iii) tant qu'il existe des sommets inexplorés :
Le sommet suivant à visiter est :
soit un adjacent du sommet le plus récemment exploré qui possède des adjacents inexplorés
soit un sommet inexploré quelconque, si le premier choix est infructueux.

EXEMPLE :



Sommet de départ x_1

Ordre x_1 , x_2 , x_3 , x_4, x_5

On remarque que dans le cas particulier d'un arbre, comme dans l'exemple ci-dessus, une depth-first search parcourt celui-ci "en profondeur".
L'implémentation d'une telle procédure peut être réalisée grâce à une pile dans laquelle on stocke les noeuds déjà explorés.
Son algorithme s'écrit alors :

Initialement : tous les sommets sont inexplorés
la pile est vide

Itération : 1:-s'il existe des sommets inexplorés alors choisir x_o un sommet inexploré;

-Sinon STOP

2 : Sélectionner x, adjacent de x_o inexploré;
-Si x existe aller en 3
-Sinon aller en 4 ;

3 : mettre x_o dans la pile ;
$x_o \longleftarrow x$;
aller en 2 ;

4 : - Si la pile est vide aller en 1;
- Sinon mettre le sommet de la pile dans x_o et aller en 2 .

N.B.: la hauteur de la pile est bornée supérieurement par le nombre de sommets dans le graphe à traiter.

L'analyse de la complexité de l'algorithme exposé ci-dessus donne :

On doit explorer chaque noeud et pour chacun d'eux on examine de continuer avec chacun de ses adjacents, donc n x d opérations (où d est le degré moyen des noeuds du graphe).

De ce fait, une depth-first search dans un graphe de n sommets exige $\delta(n^2)$ temps de calcul.

L'algorithme de la méthode de coloration s'énonce maintenant simplement :

(i) ordonner les noeuds du graphe à l'aide d'une depth-first search.

(ii) colorer les noeuds dans l'ordre ainsi établi.

De plus, puisqu'une depth-first search exige $\delta(n^2)$ temps de calcul et que les opérations de coloration en utilisent autant, cette méthode requiert $\delta(n^2)$ temps de calcul.

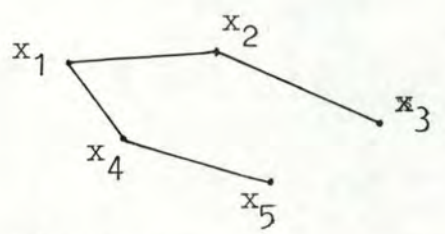
(II.1.1.6.) ORDRE BREADTH-FIRST SEARCH (/1/)

Le principe de cette méthode est de colorer les noeuds du graphe dans l'ordre où ils se présentent dans une breadth-first search .

La breadth-first search est également une méthode de parcours d'un graphe. Elle procède de la façon suivante :

A partir d'un sommet de départ, elle explore les adjacents de ce noeud, ensuite les adjacents inexplorés de ceux-ci dans l'ordre où ils ont été visités et ainsi de suite, en examinant chaque fois les adjacents des adjacents. Si, à un niveau donné de la recherche, on ne trouve plus d'adjacents inexplorés et qu'il reste des noeuds à visiter, on considère un nouveau sommet de départ et on recommence le même processus.

EXEMPLE :



Sommet de départ x_1

Ordre : x_1 , x_2 , x_4 , x_3 , x_5 .

N.B. Dans le cas particulier d'un arbre (cfr. exemple) une breadth-first search parcourt celui-ci en "largeur".

Pour l'implémentation de cette procédure on n'utilise plus une pile, comme pour la depth-first search, mais une file pour stocker les adjacents déjà explorés. L'algorithme s'écrit alors :

Initialisation : tous les sommets sont inexplorés
file vide

Itération : 1 Si il existe des sommets inexplorés alors choisir x_o noeud inexploré; ajouter x_o à la fin de la file
Sinon Stop

2 Si la file est vide aller en 1;
Sinon mettre dans x_o l'élément en début de file; aller en 3

3 Pour tout x adjacent de x_o faire
Si x inexploré alors ajouter x à la fin de la file
Fin pour
Aller en 2 ;

L'analyse de complexité de cet algorithme donne :
- les mêmes raisons que pour la depth-first search expliquent que la breadth-first search utilise $\Theta(n^2)$ temps de calcul.

L'algorithme de la méthode de coloration envisagée s'énonce alors :

(i) ordonner les noeuds du graphe à l'aide d'une breadth-first search

(ii) colorer les noeuds dans l'ordre ainsi établi.

Enfin, puisqu'une breadth-first search exige $\delta(n^2)$ temps de calcul, et que les opérations de coloration en nécessitent autant, cette méthode de coloration requiert $\delta(n^2)$ temps de calcul.

(II.1.2.) Une amélioration aux méthodes séquentielles par sommets: LA PERMUTATION BICHROMATIQUE (/1/ , /2/ , /4/)

Les méthodes exposées précédemment possèdent la particularité que lorsqu'un sommet ne peut être coloré sans utiliser une nouvelle couleur, on ne change pas les colorations déjà faites et on prend la nouvelle couleur nécessaire. La méthode de permutation par contre essaie dans ce cas de permuter des colorations faites afin de libérer une couleur pour le sommet à colorer de façon à ne pas utiliser de nouvelles couleurs.

Soit donné un graphe G ayant une k-coloration 1, 2, ..., k et G_i l'ensemble de sommets de G colorés i. Pour $i \neq j$ le sous-graphe bichromatique i,j est le sous-graphe engendré par $G_i \cup G_j$, noté $\langle G_i \cup G_j \rangle$.

Une composante connexe de $\langle G_i \cup G_j \rangle$ est dite une i,j composante . Si les couleurs i et j sont permutées sur une i, j composante du graphe k - coloré G, alors une autre k-coloration de G est obtenue. Cette procédure est appelée une $i \longleftrightarrow j$ permutation sur le graphe k-coloré G. Une permutation bichromatique sur un graphe k-coloré G est une $i \longleftrightarrow j$ permutation pour $i \neq j$.

La recherche de permutations bichromatiques dans le processus de coloration est à la base de l'algorithme suivant :

1) Ordonnancement des sommets du graphe par une des 6 méthodes étudiées précédemment.

2) Colorer x_1 avec 1 et éliminer la couleur 1 aux adjacents de x_1 ; i = 2, k = 1

3) - Si $i > n$, stop

- Sinon, soit $\langle x_1 , x_2 , \ldots x_{i-1} \rangle$ coloré 1, 2,...,k

- Si on parvient à colorer x_i avec 1, 2, ... , k colorer x_i et interdire $f(x_i)$ aux adjacents de x_i .

- Sinon, - Si pour i, j , $1 \leq i < j \leq k$, toute i, j composante de $\langle x_1 , x_2, \ldots , x_{i-1} \rangle$ a des adjacents à x_i possèdant tous la même couleur alors faire une $i \longleftrightarrow j$ permutation sur les i, j composantes de $\langle x_1 , x_2, \ldots x_{i-1} \rangle$ dont les adjacents à x_i sont colorés i . Colorer les sommets $x_1, \ldots, x_{i-1}$ de même que dans cette nouvelle coloration et x_i avec la couleur i, et une k-coloration de $\langle x_1 , x_2, \ldots x_i \rangle$ est obtenue; interdire i aux adjacents de x_i .

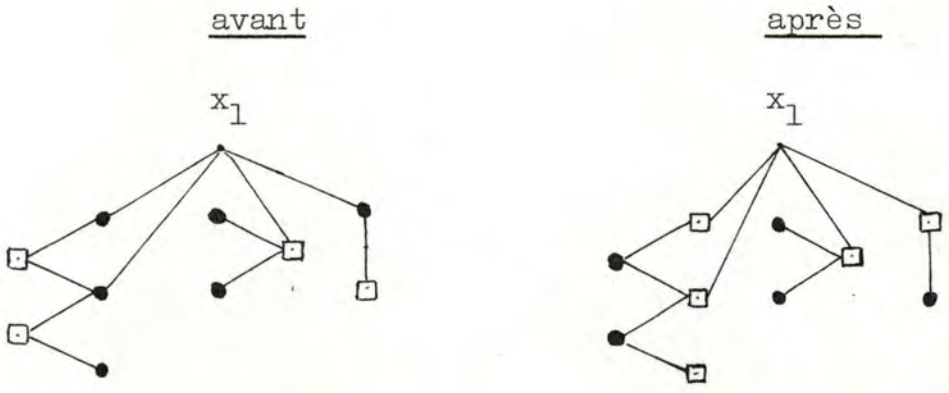
- Sinon, une telle permutation n'est pas possible; colorer $x_1, \ldots, x_{i-1}$ de même qu'en $\langle x_1, \ldots x_{i-1} \rangle$ et colorer x_i avec k + 1 ; interdire k + 1 aux adjacents de x_i . Ainsi une k + 1 - coloration de $\langle x_1, \ldots, x_i \rangle$ est obtenue; k = k + 1.

4) i = i + 1 aller en 3)

Etudions le principe d'une permutation bichromatique sur un exemple :

Soit x_1 le sommet à colorer ne pouvant ni prendre la couleur i (notée par •), ni la couleur j (notée par □); de plus x_1 est adjacent à un seul type de sommets(• ou □) par composante,

donc une permutation bichromatique est possible, il suffit en effet d'échanger les couleurs des sommets d'une série de composantes connexes pour que x_1 ne soit plus adjacent qu'à un seul type de sommets et puisse prendre l'autre couleur.



On obtient alors plusieurs méthodes de coloration différentes si dans l'étape 1) de l'algorithme, on applique les six méthodes d'ordonnancement présentées dans (I.1.1) :

- méthode RNDI (Random-interchange)
- méthode LFI (largest-first-interchange)
- méthode SLI (smallest-last-interchange)
- méthode DGENI (DEGRE-GENERALISE-INTERCHANGE)
- méthode DFSI (DEPTH-FIRST-SEARCH-INTERCHANGE)
- méthode BFSI (BREADTH-FIRST-SEARCH-INTERCHANGE)

L'analyse de complexité de ces méthodes donne :

(i) l'ordonnancement, comme nous l'avons vu en (I.1.1) utilise au plus $\delta(n^2)$ temps de calcul.

(ii) D'autre part l'algorithme de la permutation bichromatique comporte une boucle (2), 3), 4)) que l'on exécute (n - 1) fois. Lors de chacune de ses exécutions on procède au plus à C_k^2 décompositions d'un graphe en composantes connexes (où k est le nombre de couleurs utilisées).

La décomposition d'un graphe en composantes connexes peut être réalisée à l'aide d'une depth-first search et donc exige $\delta(n^2)$ temps de calcul. Cette procédure étant visiblement la plus coûteuse parmi les opérations de la boucle, l'algorithme requiert :

$$(n - 1) \cdot \frac{k\,(k - 1)}{2} \cdot \delta(n^2) \text{ temps calcul}$$

Donc, la complexité de ces méthodes varie entre $\delta(n^3)$ et $\delta(n^4)$ suivant le nombre de couleurs utilisées.

N.B. : Si l'on peut espérer que l'approximation du nombre chromatique obtenue par ces méthodes soit meilleure que celle fournie par les méthodes du paragraphe précédent, ceci est toutefois atteint au détriment du temps calcul.

(II.1.3.) LES METHODES SEQUENTIELLES PAR COULEURS

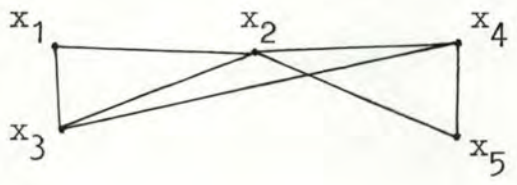
Toutes les méthodes vues jusqu'à présent donnaient une couleur à un sommet donné. Celles exposées ici, par contre, donnent des sommets à une couleur en question.

(II.1.3.1.) LARGEST-FIRST PAR COULEURS (/7/)

La stratégie utilisée par cette méthode pour attribuer des sommets à une couleur donnée est la suivante :

En supposant les noeuds ordonnés par degrés décroissants, on examine pour chaque sommet, considéré dans cet ordre, la possibilité de lui donner la couleur en question.

EXEMPLE :



ordre par degrés décroissants : x_2 , x_3 , x_4 , x_1 , x_5

colorés en 1 : x_2

colorés en 2 : x_3 , x_5

colorés en 3 : x_4 , x_1

D'où l'algorithme de coloration d'un graphe G (X, E) :

1) UNCOLORED = X ; I = 1, COLORED [i] = $\emptyset$, $1 \leqslant i \leqslant \#X$
2) SI UNCOLORED = $\emptyset$; STOP et imprimer COLORED [i], $1 \leqslant i \leqslant I$.
3) Si chaque noeud de UNCOLORED est connecté à un noeud de COLORED [I] alors I = I + 1
4) Soit x le noeud de UNCOLORED de degré maximum parmi ceux qui ne sont connectés à aucun noeud de COLORED [I] .
5) COLORED [I] = COLORED [I] U $\{x\}$,UNCOLORED = UNCOLORED - $\{x\}$, et aller en 2).

L'analyse de complexité de cet algorithme donne :

(i) la boucle 2) - 3) - 4) - 5) doit être exécutée n fois (avec $n = \#X$)

(ii) Chaque passage dans cette boucle implique(en 4)) une sélection du noeud de degré maximum, donc exige au plus n comparaisons.

(iii) D'après ces considérations, la méthode requiert $\delta(n^2)$ temps de calcul.

(II.1.3.2.) METHODE DE L'ENSEMBLE MAXIMAL INDEPENDANT (/1/ , /7/)

Comme son nom l'indique cette méthode est organisée autour du concept d'ensemble maximal indépendant d'un graphe (cfr. préliminaires). De plus, elle est basée sur le théorème suivant :

Théorème : Si un graphe est r-chromatique (i.e. son nombre chromatique est r), il peut être coloré par r couleurs en colorant d'abord un ensemble indépendant maximal m_i , puis en colorant par la couleur suivante un ensemble indépendant maximal de $G - m_i$, etc., jusqu'à ce que tous les sommets soient colorés.

Preuve : Le fait qu'une telle coloration existe toujours (utilisant r couleurs) peut être prouvé de la façon suivante. Supposons qu'une coloration de r couleurs existe t.q. un ou plusieurs ensembles de même couleur ne seront pas des ensembles indépendants maximaux. Par numérotation des couleurs de façon quelconque, on peut alors toujours colorer avec la couleur 1 les sommets $\overline{X}_1$ non colorés 1 qui forment un ensemble indépendant maximal avec les sommets X_1 colorés 1. Cette nouvelle coloration est possible, vu que l'ensemble des sommets de $\overline{X}_1$ est non-adjacent aux sommets de X_1 ; donc les sommets adjacents à $\overline{X}_1$ ont des couleurs différentes de 1 et ne sont par conséquent pas affectés par l'échange de la couleur des sommets de $\overline{X}_1$. Considérons maintenant le sous-graphe $< X - (X_1 \cup \overline{X}_1) >$; la même procédure mènera alors vers une coloration contenant un ensemble indépendant maximal coloré 2, etc...

Malheureusement, ce théorème ne précise pas quel ensemble indépendant maximal doit être coloré à chaque étape, ainsi n'importe quel ensemble indépendant maximal peut être considéré. La solution proposée par la méthode présentée est de choisir à chaque étape un ensemble indépendant maximal de cardinalité maximale.

D'où l'algorithme MIS

1) I = 1 ;

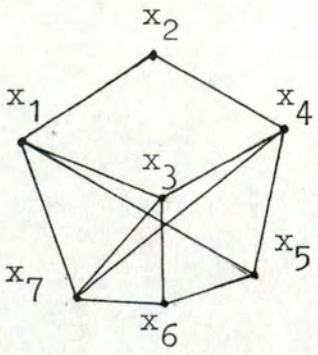
2) Soit U l'ensemble de tous les noeuds non colorés.
   Si U = $\emptyset$ stop

3) Soit U_I un ensemble indépendant de cardinalité maximum de U

4) $\forall$ u $\in$ U_I : colorer u avec I ;
   I = I + 1 ; aller en 2)

Exemple de fonctionnement :

Soit le graphe suivant à colorer



1) I = 1

2) U = $\{ x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 \}$

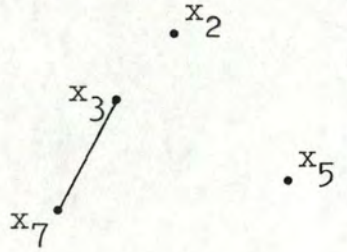
3) les ensembles indépendants maximaux sont :

$\{x_1, x_4, x_6\}$; $\{x_2, x_5, x_7\}$; $\{x_3, x_5, x_2\}$; $\{x_2, x_6\}$

$\Rightarrow U_I = \{x_1, x_4, x_6\}$

4) colorer x_1, x_4, x_6 avec la couleur 1 ; I = 2 ; aller en 2)

2) U = $\{ x_2, x_3, x_5, x_7 \}$



3) les ensembles indépendants maximaux sont :

$$\{x_2, x_5, x_7\} \; ; \; \{x_2, x_3, x_5\} \Rightarrow U_I = \{x_2, x_5, x_7\}$$

4) colorer x_2, x_5, x_7 avec la couleur 2 ; I = 3; aller en 2)

2) $U = \{x_3\}$

3) $U_I = \{x_3\}$

4) colorer x_3 avec la couleur 3 ; I = 4 ; aller en 2)

2) $U = \emptyset$ stop avec la coloration 1, 2, 3, 1, 2, 1, 2.

Cette algorithme a cependant un sérieux inconvénient. Le problème de trouver un ensemble indépendant de taille maximum dans un graphe, qui doit être résolu chaque fois que l'étape 4) doit être exécutée, ne peut être implémenté en un temps de calcul polynomial. Donc, le temps de calcul requis par la méthode elle-même, croît, avec la taille du problème, plus rapidement que n^{ε} , où n est le nombre de noeuds du graphe et $\varepsilon > 0$ quelconque. Ceci implique que l'application de cette méthode sur des graphes importants est quasiment irréalisable faute de temps. Un algorithme pour trouver les ensembles indépendants maximaux d'un graphe sera exposé au chapitre III.

Cependant, il existe un algorithme qui ne comporte pas cet inconvénient et qui "approxime" celui présenté dans ce paragraphe :

(II.1.3.3.) METHODE "APPROXIMATELY MAXIMUM INDEPENDENT SET" (/1/, /2/, /7/)

L'algorithme AMIS est le suivant :

1) I = 1 ;

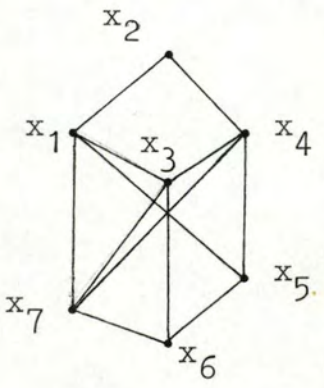
2) Soit U l'ensemble de tous les sommets non colorés
Si $U = \emptyset$; stop

3) Soit x le noeud de degré minimum dans le sous-graphe induit par U . Colorer x avec I et $U = U - (\{x\} \cup \{y \in X ; \langle x,y \rangle \in E\})$.

4) Si $U = \emptyset$; $I = I + 1$ aller en 2)
Sinon aller en 3)

Exemple de fonctionnement

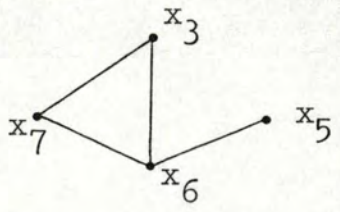
Considérons le graphe de l'exemple précédent.



1) $I = 1$

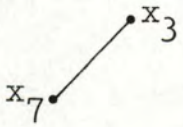
2) $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$

3) Soit x_2 le noeud de degré minimum; colorer x_2 avec 1
$U = \{x_3, x_5, x_6, x_7\}$



4) aller en 3)

3) Soit x_5 le sommet de degré minimum; ; colorer x_5 avec 1 ;
$U = \{x_3 , x_7\}$

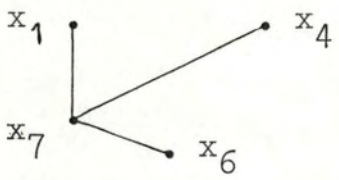


4) aller en 3)

3) soit x_3 le sommet de degré minimum ; colorer x_3 avec 1 ;
$U = \emptyset$

4) $U = \emptyset$ => I = 2 ; aller en 2)

2) $U = \{x_1, x_4, x_6, x_7\}$



3) soit x_1 le sommet de degré minimum ; colorer x_1 avec 2 ;
$U = \{x_4, x_6\}$

• x_4

• x_6

4) aller en 3)

3) soit x_4 le sommet de degré minimum; colorer x_4 avec 2 ;
$U = \{x_6\}$

4) aller en 3)

3) colorer x_6 avec 2 ; $U = \emptyset$

4) $U = \emptyset$ => I = 3 ; aller en 2)

2) $U = \{x_7\}$

3) colorer x_7 avec 3 ; $U = \emptyset$

4) $U = \emptyset$ => I = 4 ; aller en 2)

2) $U = \emptyset$ => STOP avec la coloration 2, 1, 1, 2, 1, 2, 3

L'analyse de complexité donne :

Comme, d'une part, on exécutera n fois l'étape 3), et d'autre part, celle-ci exige le calcul du degré des noeuds de U, donc au plus n^2 opérations, et la sélection du noeud de degré minimum, donc au plus n comparaisons, cette méthode requiert $\Theta(n^3)$ temps de calcul.

(II.2.) LES METHODES DE SATURATION (/8/)

Les méthodes séquentielles par sommets ordonnent d'abord les sommets avant de les colorer dans l'ordre établi.
Les méthodes de saturation par contre réordonnent les sommets non colorés après coloration d'un sommet quelconque en utilisant les informations relatives aux colorations déjà faites.

Soit le graphe G (X,E) ; définissons le degré de saturation d'un sommet x, noté dsat (x) comme étant le nombre de couleurs adjacentes c-à-d. le nombre de couleurs telles qu'il existe au moins un sommet adjacent à x possédant cette couleur.

Donc, $dsat(x) = \#\{ h \text{ tq } \exists y \in X ; (x,y) \in E , f(y) = h \}$

Le principe de l'algorithme dit de saturation est :
après coloration d'un sommet quelconque, prendre comme sommet suivant à colorer celui dont le degré de saturation est le plus élevé parmi les sommets non colorés.

d'où l'algorithme SAT :

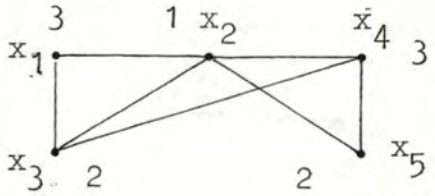
1) Ordonner tous les sommets selon la notion de degré maximal (LF) c-à-d. $d1 \geq d2 \ldots \geq dn$
2) Colorer le sommet x_1 avec la première couleur ; Interdire cette couleur pour les adjacents de x_1 .
3) Prendre le sommet dont le degré de saturation est le plus élevé parmi les sommets non colorés (en cas d'égalité utiliser l'ordre 1)).

4) Colorer ce sommet avec la première couleur possible et interdire cette couleur à ses adjacents.

5) S'il reste des sommets à colorer, aller en 3) . Sinon arrêter.

Exemple de fonctionnement

Soit le graphe suivant à colorer :



1) ordre par degrés décroissants x_2, x_3, x_4, x_1, x_5

2) colorer x_2 avec la couleur 1 .
Interdire la couleur 1 à x_1, x_3, x_4, x_5

3) tous les sommets non colorés ont même degré de saturation
$\Rightarrow$ on sélectionne le sommet x_3 (cfr 1))

4) colorer x_3 avec 2; interdire 2 à x_1, x_2 , x_4

5) aller en 3)

3) le degré de saturation de x_1 et x_4 vaut 2 ; celui de x_5 vaut 1 ; donc on sélectionne x_4

4) colorer x_4 avec 3; interdire 3 à x_2, x_3, x_5 ;

5) aller en 3)

3) le degré de saturation de x_1 et x_5 vaut 2 ; donc on sélectionne x_1

4) colorer x_1 avec 3 ; interdire 3 à x_2, x_3 ;

5) aller en 3)

3) sélection de x_5

4) colorer x_5 avec 2, interdire 2 à x_2, x_4 ;

5) STOP avec la coloration 3, 1, 2, 3, 2 .

Etudions un résultat théorique de cet algorithme

Théorème : L'algorithme SAT donne la valeur exacte de $\chi(G)$ pour les graphes bipartis ($\chi(G) = 2$)

Rappelons qu'un graphe biparti peut être partitionné en 2 classes X_1 et X_2 de telle sorte que deux noeuds de la même classe ne soient jamais adjacents.

Démonstration :

Soit G un graphe biparti connexe avec au moins 3 sommets. Supposons qu'un noeud x a un degré de saturation égal à 2 ; dans ce cas, x a deux adjacents à couleurs distinctes et on peut construire deux chaînes. Comme G est fini, on peut trouver un sommet commun y des deux chaînes, donc un cycle. Alors, ou bien le cycle est pair (nombre pair d'arêtes) et les deux adjacents sont de même couleur ou bien G n'est pas biparti.
L'algorithme SAT n'utilise par conséquent pas plus de deux couleurs.

La complexité de l'algorithme SAT s'évalue de la manière suivante :

> On effectue (n - 1) fois la boucle 3), 4), 5) ; à chacune de ces exécutions, on détermine le sommet dont le degré de saturation est le plus élevé parmi les sommets non colorés, ce qui entraîne au plus n comparaisons. Puisque cette opération est la plus coûteuse en temps, la méthode requiert $\mathcal{O}(n^2)$ temps calcul.

Puisque la permutation bichromatique étudiée en (II.1.2) peut être utilisée pour n'importe quel algorithme de coloration des noeuds, par la plus petite couleur possible, dans un ordre donné nous pouvons l'appliquer pour l'algorithme SAT : il suffit de remplacer l'étape 4) de SAT par l'étape 3) de la permutation bichromatique.

Le nouvel algorithme ainsi obtenu sera appelé

DSI : Degré-Saturation-Interchange

Des considérations faites à propos du temps de calcul requis par l'algorithme de permutation bichromatique on déduit que la complexité de DSI est de l'ordre de $\mathcal{O}(n^3)$ à $\mathcal{O}(n^4)$ suivant le nombre de couleurs utilisées.

(II.3.) METHODES DIVERSES NON-EXACTES

Les méthodes présentées dans ce chapitre, ne fournissent également qu'une approximation du nombre chromatique, mais se distinguent de celles vues précédemment de par leur conception qui est plus sophistiquée .

(II.3.1.) METHODE "ADJACENTS DES ADJACENTS" (/9/)

La plupart des méthodes de coloration exposées jusqu'ici utilisent la notion de degré des sommets. Or, cette notion ne nous donne qu'une information très relative sur la structure du graphe, laquelle détermine le nombre chromatique. La structure elle-même se caractérise par l'assemblage des arêtes entre elles et non pas par le nombre d'arêtes passant par un sommet (degré).

Cette remarque suggère la recherche d'un algorithme qui ne serait pas basé uniquement sur l'aspect global, mais aussi- et davantage - sur la structure. Nous définissons un algorithme utilisant la méthode appelée "ADJACENTS DES ADJACENTS".

L'algorithme AA est le suivant :
soit le graphe G (X,E) à n sommets ;

1) i = 1 ; G1 = G ; n1 = n (à l'étape i, on cherche les sommets de couleur i dans le graphe G_i) ; k_o = o

2) $k = k_o$ et $j = 1$, chercher un sommet x_o de degré maximum dans G_i

3) $k = k + 1$, $A_{j-1} = \{x_o\}$. x_o prend la couleur i

4) chercher A_j = ensemble des adjacents de x_o , poser $k = k + \# A_j$

5) Si $k = n_i$ aller à 9) sinon aller à 6)

6) chercher A_{j+1} = ensemble des adjacents, non contenus dans A_{j-1} et A_j , des sommets appartenant à A_j

7) Si $A_{j+1} = \emptyset$, chercher un sommet x_o de degré maximum parmi les sommets restants (non contenus dans $\cup A_j$), poser $j = j+3$ et aller à 3) (en effet le graphe G_i n'étant pas connexe dans ce cas, il faut refaire les mêmes opérations sur une autre composante connexe).
Sinon, poser $k = k + \# A_{j+1}$ et colorer avec la couleur i, autant qu'il est possible, les sommets contenus dans A_{j+1} dans l'ordre décroissant des degrés dans G_i

8) Si $k = n_i$ aller à 9), sinon, poser $j = j+1$ et aller à 6)

9) Imprimer les sommets de couleur i, poser $i = i+1$, chercher le sous-graphe restant (G_i, n_i)

10) Si tous les sommets de G_i sont isolés, ils prennent alors la couleur i.
Les imprimer ainsi que le nombre chromatique $\chi = i$ et s'arrêter .
Sinon aller en 2) en colorant déjà les sommets isolés de G_i , s'il en existe, avec la couleur i (leur nombre sera placé dans k_o)

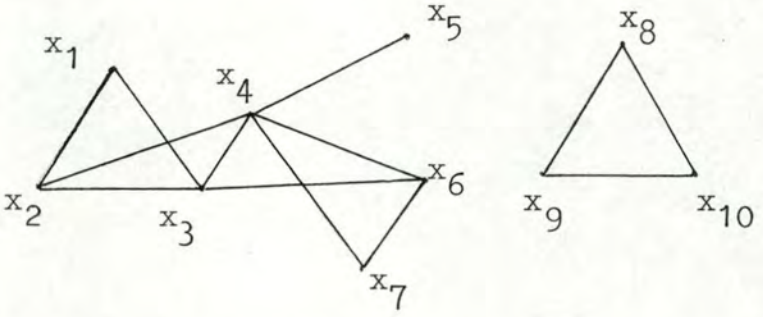
Remarques :

a) cet algorithme tient compte des liaisons ou assemblages des arêtes tandis que l'algorithme LF suit un chemin uniquement en fonction des degrés, c-à-d. que l'on va d'un bout à l'autre du graphe sans tenir compte des liaisons entre les sommets choisis; ceci peut augmenter le nombre chromatique cherché.

b) à l'étape i, le degré d'un sommet est calculé dans le sous-graphe restant G_i et non pas dans le graphe de départ G.

Exemple de fonctionnement

Nous étudions sur un exemple concret le fonctionnement de l'algorithme; soit le graphe à 10 sommets :



$G_1 (X_1 , E_1) = G (X,E)$ $\quad X_1 = X$

$E_1 = E$

1) $i = 1 ; n1 = 10$

2) $k = o, \ j = 1 ; \ x_o = x_4 ;$

3) $k = 1 , \ Ao = \{x_4\}$; x_4 prend la couleur 1

4) $A_1 = \{x_2 , x_3, x_5 , x_6, x_7\} ; \ \#A_1 = 5 ; \ k = 6$

5) aller en 6)

6) $A_2 = \{x_1\}$

7) $k = 7$; x_1 prend la couleur 1

8) $j = 2$; aller en 6)

6) $A_3 = \emptyset$

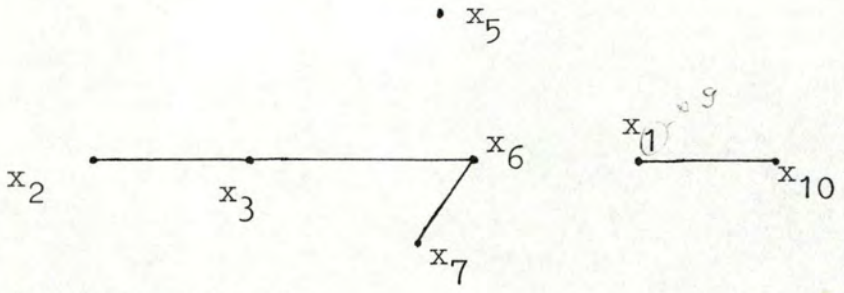
7) soit x_8 de degré maximal, $j = 5$, aller en 3)

3) $k = 8$, $A_4 = \{x_8\}$; x_8 coloré avec 1

4) $A_5 = \{x_9, x_{10}\}$; $\#A_5 = 2$; $k = 10$

5) aller en 9)

9) $i = 2$; $G_2 (X_2 , E_2)$; $n_2 = 7$



10) colorer x_5 avec 2

2) $k = 1$, $j = 1$, $x_o = x_3$

3) $k = 2$, $A_o = \{x_3\}$, colorer x_3 avec 2

4) $A_1 = \{x_2 , x_6\}$; $\#A_1 = 2$, $k = 4$

5) aller en 6)

6) $A_2 = \{x_7\}$

7) $k = 5$; colorer x_7 avec 2

8) $j = 2$; aller en 6)

6) $A_3 = \emptyset$

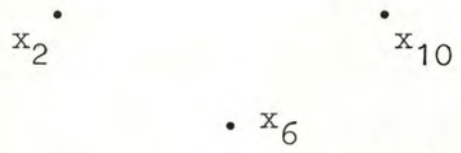
7) soit x_9 de degré maximal ; j = 5 , aller en 3)

3) k = 6 , $A_4 = \{ x_9 \}$, colorer x_9 avec 2

4) $A_5 = \{ x_{10} \}$ $\#A_5 = 1$ k = 7

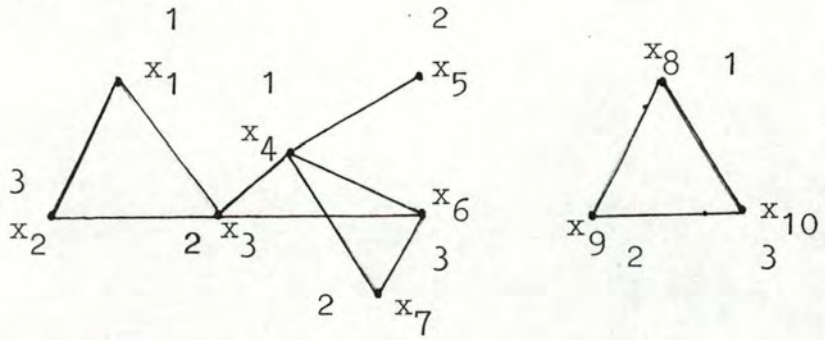
5) aller en 9)

9) i = 3 $G_3 (X_3 , E_3)$ $n_3 = 3$



10) colorer x_2, x_6, x_{10} avec 3 , STOP

D'où la coloration



Analyse de la complexité

Un sommet coloré avec la couleur i devra être stocké i fois dans 1 A_j.De plus, pour chaque noeud contenu dans A_j , on examine, pour chacun de ses adjacents, la possibilité d'appartenir à A_{j+1} . Ce dernier test exige n × d opérations (où d est le degré moyen d'un noeud du graphe). Donc si n_i représente le nombre de sommets colorés avec la couleur i et k le nombre total de couleurs, la somme des temps calcul des différentes exécutions de l'étape 6 s'élève à

$$\sum_{i=1}^{k} n_i \cdot i \cdot \Theta(n \cdot d) \leq k \cdot \sum_{i=1}^{k} n_i \cdot \Theta(n \cdot d) = \Theta(k \cdot n^2 \cdot d)$$

Comme l'étape 6 est la plus coûteuse en temps calcul, la complexité de l'algorithme étudié est $\Theta(n^4)$. Toutefois, dans le chapitre IV, nous présenterons une implémentation de la méthode "adjacents des adjacents" dont la dépendance vis-à-vis du temps est de l'ordre de $\Theta(n^2)$ pour un graphe vérifiant $k \times d \simeq n$. Cette dernière propriété est vérifiée par beaucoup de graphes rencontrés dans des applications pratiques.

(II.3.2.) METHODE "LARGEST FIRST RECURSIF (RLF)" (/2/)

L'algorithme RLF combine la stratégie de LF avec la structure de l'algorithme AMIS. Comme dans la méthode LF, à chaque étape de la procédure RLF, on sélectionne un noeud, pour le colorer, qui, dans un certain sens, permettra de colorer les noeuds restants avec la plus petite couleur possible. Comme la méthode AMIS, la procédure RLF termine l'attribution de la couleur i avant de commencer l'attribution de la couleur $i + 1$.

Soit donné un graphe G (X,E) alors l'algorithme RLF suit le raisonnement suivant :

* assigner la couleur 1 au sommet de degré maximal de G, soit x_1
* supposons que i sommets soient colorés avec la couleur 1
  . alors, choisir, si possible, $x_{i+1} \in X_1$ t.q. $d_{X_2}(x_{i+1}) = \max \left\{ d_{X_2}(x_j) \mid x_j \in X_1 \right\}$; où X_1 est l'ensemble des sommets non colorés, non-adjacents à tout sommet coloré et X_2 est l'ensemble des sommets non colorés adjacents à au moins un sommet coloré. Si on obtient plusieurs candidats pour x_{i+1} , on prend le sommet de degré minimal sur X_1 .

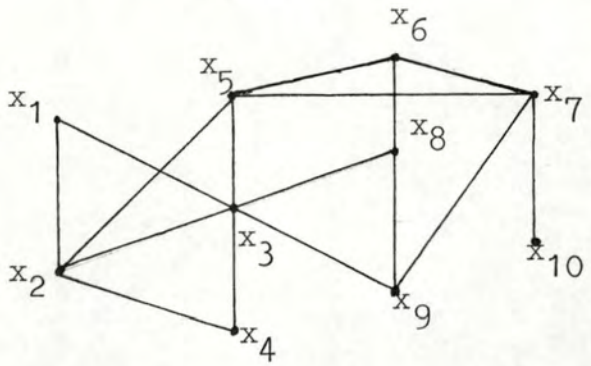
* Si X_1 est vide, alors répéter le processus sur le sous-graphe engendré par les sommets non colorés de G en utilisant la couleur suivante.

* Ce processus recursif est répété jusqu'à ce que tous les sommets de G soient colorés.

Exemple :

. Soit le graphe suivant G (X,E)

n = 10



les degrés des noeuds sont : d (x_1) = 2 ; d (x_2) = 4 ; d (x_3) = 6 ; d(x_4) = 2 ; d (x_5) = 4 ; d (x_6) = 3 ; d (x_7) = 4 ; d(x_8) = 3 ; d (x_9) = 3 ; d (x_{10}) = 1 .

* le sommet de degré maximum est x_3 ⇒ colorer x_3 avec 1

D'où $X_1 = \{ x_6, x_7, x_{10} \}$; $X_2 = \{ x_1, x_2, x_4, x_5, x_8, x_9 \}$

et $d_{X_2}(x_6) = 2$ $d_{X_2}(x_7) = 2$ $d_{X_2}(x_{10}) = 0$

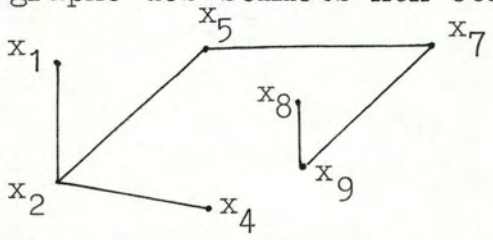
⇒ Colorer x_6 avec 1

D'où $X_1 = \{ x_{10} \}$ $X_2 = \{ x_1, x_2, x_5, x_8, x_9, x_7 \}$ et $d_{X_2}(x_{10})$ = 1

$\Rightarrow$ Colorer x_{10} avec 1

D'où $X_1 = \emptyset$ et il faut recommencer le processus sur le sous-graphe restant G_1 .

* Le sous-graphe des sommets non colorés $G_1\ (X_1, E_1)$ est :



$d_{G_1}(x_1) = 1 \quad d_{G_1}(x_2) = 3 \quad d_{G_1}(x_5) = 2 \quad d_{G_1}(x_7) = 2$

$d_{G_1}(x_8) = 1 \quad d_{G_1}(x_9) = 2 \quad d_{G_1}(x_4) = 1$

$\Rightarrow$ Colorer x_2 avec 2

D'où $X_1 = \{x_7, x_8, x_9\}$ $X_2 = \{x_1, x_4, x_5\}$ et

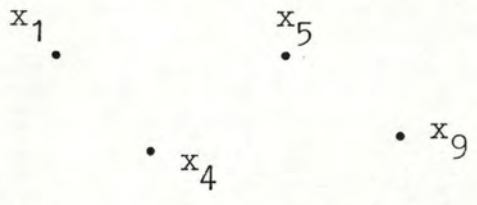
$d_{X_2}(x_7) = 1 \quad d_{X_2}(x_8) = 0 \quad d_{X_2}(x_9) = 0$

$\Rightarrow$ Colorer x_7 avec 2

D'où $X_1 = \{x_8\}$, $X_2 = \{x_1, x_4, x_5, x_9\}$ et $d_{X_2}(x_8) = 1$

$\Rightarrow$ Colorer x_8 ave 2 ; $X_1 = \emptyset$, recommencer sur le sous-graphe restant G_2

* Sous-graphe des sommets non colorés $G_2\ (X_2, E_2)$



$d_{G_2}(x_1) = d_{G_2}(x_4) = d_{G_2}(x_5) = d_{G_2}(x_9) = 0$

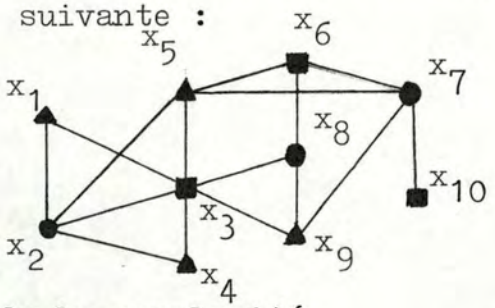
$\Rightarrow$ Colorer x_1 avec 3

D'où $X_1 = \{x_4, x_5, x_9\}$ $X_2 = \emptyset$ et $d_{X_2}(x_4) = d_{X_2}(x_5) = d_{X_2}(x_9) = 0$

et $d_{X_1}(x_4) = d_{X_1}(x_5) = d_{X_1}(x_9) = 0$

$\Rightarrow$ Colorer x_4 , x_5 , x_9 avec 3

$X_1 = \emptyset$ et tous les sommets sont colorés de la façon suivante :



couleurs 1 $\Leftrightarrow$ ■
2 $\Leftrightarrow$ ●
3 $\Leftrightarrow$ ▲

Analyse de la complexité

Quand on colore un sommet x_i, il faut transférer tous ses adjacents non colorés dans X_2 et, pour chaque noeud x, adjacent de ces derniers, il faut modifier $d_{X_2}(x)$.

Puisque cette opération exige $\Theta(n^2)$ temps calcul, la complexité de l'algorithme RLF vaut $\Theta(n^3)$. Comme la méthode "adjacents des adjacents", cependant, la procédure RLF requiert seulement $\Theta(n^2)$ temps calcul pour colorer des graphes pour lesquels $k.d \simeq n$ où k est le nombre de couleurs utilisées pour colorer le graphe et d est le degré moyen d'un noeud. (cfr. chapitre IV pour la preuve).

(II.3.3.) METHODE DE "L'ENUMERATION IMPLICITE APPROXIMATIVE" (/ 10 /)

Le principe de cette méthode est le suivant :

Etablir une coloration provisoire au moyen d'une méthode séquentielle ce qui donnera une première borne supérieure au nombre chromatique.
Cette coloration est alors améliorée en essayant de diminuer le nombre de couleurs utilisées. La stratégie de base est le retrait de couleur : pour changer la couleur d'un sommet quelconque en une couleur inférieure, il suffit de recolorer un adjacent du sommet en question.

Définissons la matrice de temps L d'un problème de coloration de graphe de la manière suivante :

$L(i,h) = 0$ si le sommet i ne peut être coloré h
$\quad\quad\quad 1$ sinon

Notons, maintenant, L^i la matrice de temps après la coloration du sommet x_i et L^o la matrice de temps avant le début de la coloration.

L'algorithme ENIMPA s'écrit alors :

Soit le graphe G (X,E) à colorer et f la fonction de coloration.

1) Ordonner les sommets selon un critère connu (cfr.(II.1.1.))
   $q_o = n$

2) Colorer x_1 : - $f(x_1) = j$ ou $j = \min \left\{ j' \text{ t.q. } L^o(1,j') = 1 \right\}$;
   $L^1 = L^o$
   - $L^1(i',j) = 0 \; \forall \; x_i$, adjacent à X_1

3) Colorer chacun des sommets x_i restants avec la plus petite couleur possible :

- $f(x_i) = j$ où $j = \min \{ j' \text{ t q } L^{i-1}(i,j') = 1 \}$

- $L^i = L^{i-1}$, $L^i(i',j)=0 \ \forall x_{i'}$ adjacent à x_i

4) Soit $q = \max \{ j \text{ t.q. } \exists x_i , f(x_i) = j \}$;

Si $q < q_o$ mémoriser la coloration; $q_o = q$

Nous allons dans la suite essayer de trouver une coloration utilisant moins de q couleurs.

5) $x_{i^o} = \min_X \{ x_i \text{ t.q. } f(x_i) = q \}$

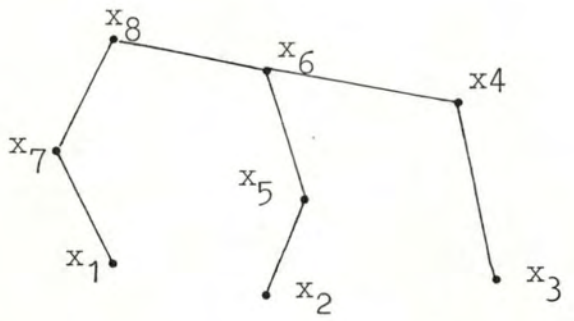
6) Si $(L^o(i^o,j) = o \ \forall \ j < q) \vee (i^o = 1)$ alors STOP

Soit $x_K = \max_X \{ x_i \text{ t.q. } i < i^o \wedge (i,i^o) \in E \}$;

- si x_k n'existe pas alors STOP ;
- Sinon soit $j_k = f(x_k)$, $j'_k = \min \{ j' \text{ t q } (L^{k-1}(k,j')=1 \wedge (j' \geqslant j_k+1) \}$
  - si $(j'_k \geq q) \vee (j_k'$ n'existe pas), alors $x_{i^o} = x_k$

    aller en 6)
  - sinon colorer x_k avec j'_k en appliquant le processus 2), utilisant la matrice de temps L^{k-1} ce qui donnera L^k ; recolorer x_{k+1} ..., x_n en appliquant le processus 3) à L^k : ceci fournit les matrices de temps L^{k+1}, ..., L^n , aller en 4)

Exemple



Supposons que l'ordre de coloration fournit par le critère de l'étape 1) est x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8

Les étapes 2), 3) colorent les sommets x_1, ..., x_8 avec 1, 1, 1, 2, 2, 1, 2, 3

4) $q = 3$

5) $i^o = 8$

6) $x_k = x_7$ et $j_k = 2$, donc j_k' n'existe pas $\Rightarrow x_{i^o} = x_7$ aller en 6)

6) $x_k = x_1$ et $j_k = 1$; $j_k' = 2$ et la nouvelle coloration est 2, 1, 1, 2, 2, 1, 1, 2

4) $q = 2$

5) $i^o = 1 \Rightarrow$ STOP.

Analyse de la complexité

Soit un sommet x_{i^o} t q $f(x_{i^o}) = q = \max \{ j \text{ t q } \exists x_i \ f(x_i) = j$

Considérons aussi la suite $x_{k_1}, \ldots, x_{k_m}$ définie de la manière suivante :

$$x_{k_1} = \max \{ x_i \text{ t q } i < i^o \wedge (i, i^o) \in E \}$$

$$x_{k_2} = \max \{ x_i \text{ t q } i < k_1 \wedge (i, k_1) \in E \}$$

$$\vdots$$

$$x_{k_m} = \max \{ x_i \text{ t q } i < k_{m-1} \wedge (i, k_{m-1}) \in E \}$$

$\exists x_i$ t q $i < k_m \wedge (i, k_m) \in E$

Notons que $m < n - 1$

Soit k le nombre moyen de couleurs $< q$ que les sommets $x_{k_1}, \ldots x_{k_m}$ peuvent encore prendre.

Alors le nombre de colorations que l'algorithme peut générer pour tenter de diminuer la couleur x_{i_o} est k^m

D'autre part, puisque $k < q$, $m < n$ et qu'une coloration par le processus 3 requiert $\delta(n^2)$ temps de calcul, la complexité de la méthode est de l'ordre de $\delta(q^n n^2)$.

Donc, cette méthode ne peut malheureusement pas être implémentée en un temps de calcul polynomial.

(II.4.) RESULTATS DE TESTS

Les différentes méthodes présentées dans ce chapitre ont fait l'objet d'une implémentation en FORTRAN 77 sur un ordinateur VAX/VMS. Ceci avait pour but de tester ces algorithmes de coloration sur 80 graphes, générés aléatoirement, dont le nombre de sommets n, la densité μ et le nombre chromatique χ étaient connus (pour le générateur aléatoire de graphes - voir en Annexe 1). Le fait que le nombre chromatique des graphes qui constituent les jeux de test soit connu, permet de mesurer la qualité des colorations fournies par les différentes méthodes.
Ces 80 graphes sont répartis également dans les 16 classes suivantes :

N° de classe	Nbre de sommets	Densité	Nbre chromatique
I	20	0.2	4
II	20	0.4	5
III	20	0.6	5
IV	20	0.8	10
V	50	0.2	5
VI	50	0.4	10
VII	50	0.6	10
VIII	50	0.8	25
IX	75	0.2	5
X	75	0.4	10
XI	75	0.6	15
XII	75	0.8	25
XIII	100	0.2	5
XIV	100	0.4	10
XV	100	0.6	25
XVI	100	0.8	50

Les performances des méthodes de coloration sont caractérisées par 2 mesures.

i) la moyenne, sur une classe, du nombre de couleurs utilisées

ii) la moyenne, sur une classe, du temps CPU consommé

Le tableau de la figure (II.1) reprend les résultats obtenus pour les classes II, VI, X, XIV (pour les résultats correspondants aux autres classes voir en annexe 2).

Les graphiques des figures (II.2) et(II.3) montrent, respectivement, l'évolution du temps de calcul et du pourcentage d'erreur par rapport au nombre chromatique pour

. une méthode séquentielle par sommets : LF
. une méthode de saturation : SAT

- une méthode séquentielle par sommets munie de la permutation bichromatique : LFI
- des méthodes séquentielles par couleurs; AMIS et MIS
- les méthodes non-exactes diverses.

	CLASSE [illegible]		CLASSE [illegible]		CLASSE X		CLASSE XIV	
	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps
[illegible]	5.60	0.03	12.20	0.11	15.40	0.19	16.60	0.31
LFI	5.20	0.11	11.20	2.04	14.00	6.64	13.80	17.87
[illegible]	5.00	0.03	12.00	0.12	15.20	0.24	15.60	0.43
[illegible]	5.00	0.05	11.00	1.65	13.40	5.40	13.40	12.99
DFS	6.40	0.03	14.60	0.11	17.80	0.21	18.80	0.37
DFSI	5.60	0.20	12.40	4.42	15.00	16.99	16.00	35.24
BFS	6.40	0.02	14.80	0.08	18.20	0.18	18.20	0.28
BFSI	5.00	0.14	12.00	3.19	15.20	11.54	15.00	27.98
DGEN	6.00	0.03	12.00	0.12	15.00	0.25	16.00	0.41
DGENI	5.00	0.11	11.00	1.90	13.80	7.00	14.20	17.20
RND	7.00	0.03	14.40	0.10	17.00	0.20	18.40	0.31
RNDI	5.60	0.27	12.40	4.64	15.00	17.05	15.60	32.61
LFSC	6.00	0.01	12.00	0.04	15.00	0.08	16.00	0.14
MIS	6.20	0.55	11.40	25.26	15.60	295.95	13.40	1532.26
AMIS	7.00	0.02	14.60	0.16	18.20	0.34	18.00	0.62
SAT	5.40	0.02	11.80	0.11	14.00	0.23	14.60	0.39
DSI	5.20	0.07	11.40	1.19	13.80	4.58	13.40	10.01
AA	5.60	0.02	12.20	0.13	14.20	0.25	15.40	0.43
RLF	5.20	0.02	11.60	0.16	14.00	0.35	13.40	0.56
[illegible]	5.00	6.83	-	-	-	-	-	-

Fig. (II.1) l'unité de temps = 1 sec CPU,
'-' = plus de 2h. CPU

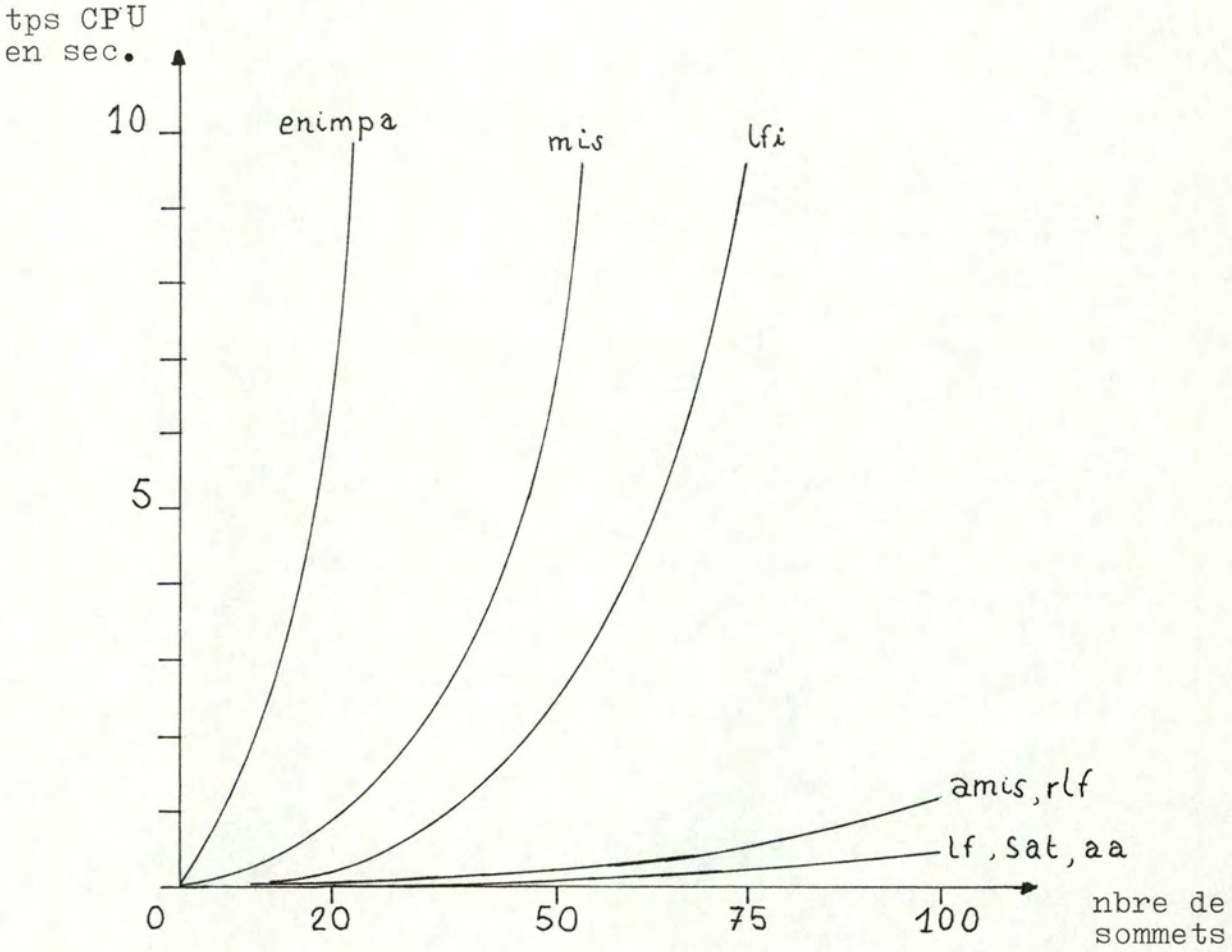


fig. (II.2)

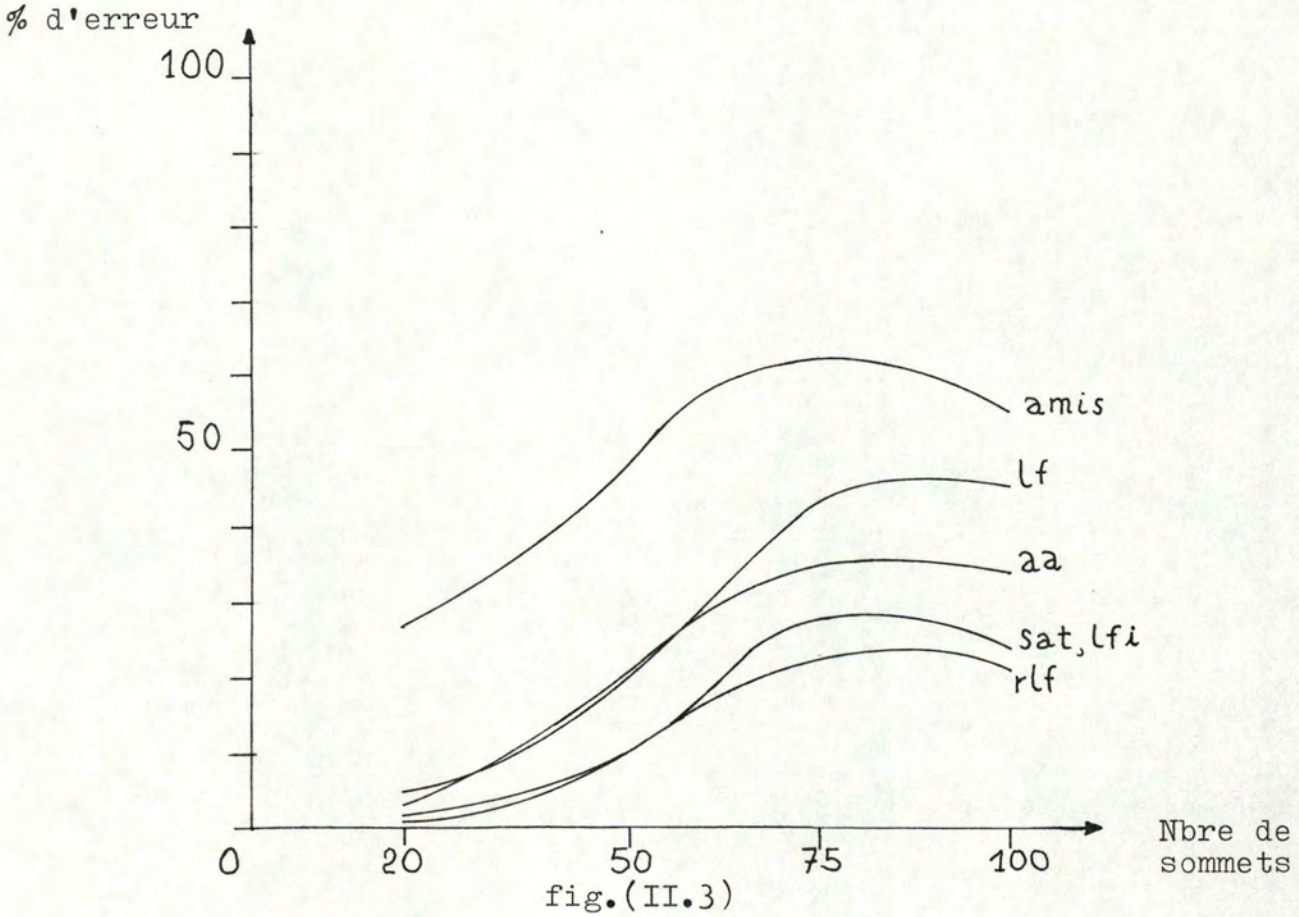


fig.(II.3)

CONCLUSIONS DES TESTS

Des tests numériques, les conclusions suivantes peuvent être tirées:

1) les méthodes de permutation améliorent les méthodes séquentielles de façon considérable : $\pm$ 10 % pour les meilleurs algorithmes; $\pm$ 17 % pour les plus mauvais.
On constate aussi l'augmentation du temps calcul, annoncée au paragraphe (II.1.2.) , que cause l'introduction de l'algorithme de permutation bichromatique dans les méthodes séquentielles par sommets.

2) Parmi les méthodes séquentielles, les méthodes BFS, DFS, RND , AMIS, sont les plus mauvaises, ce qui, pour le dernier algorithme cité, est assez surprenant suite à la lecture de / 1 / . LF et SL sont assez équivalentes, mais on note une légère supériorité pour SL. La méthode la plus rapide est LFSC; son efficacité dans la coloration est du même ordre que LF. Contrairement à ce qui avait été annoncé précédemment, DGEN n'améliore pas toujours la coloration de LF et dans les cas où cela se produit, la différence n'est guère significative.
Comme prévu, le temps calcul requis par l'algorithme MIS est le plus important, surtout pour les densités faibles. C'est malheureusement pour des grandes dimensions à faibles densités (graphes fréquemment rencontrés dans la pratique) que ce dernier algorithme impose une prestation plus longue que les autres méthodes séquentielles.

3) Remarquons les très bonnes performances des méthodes de saturation.
L'algorithme SAT peut d'ailleurs concurrencer les méthodes séquentielles munies de la permutation bichromatique.

4) Dans le chef des méthodes approximatives diverses, l'énumération implicite approximative est à éviter vu le temps de calcul requis et le fait qu'elle ne garantit qu'une approximation du nombre chromatique. Le nombre de couleurs utilisées par l'algorithme RLF est proche de celui de SAT, toutefois il s'avère que RLF est préférable à SAT pour des graphes de faibles densités, d'autant que pour ces derniers, la remarque, concernant le temps de calcul requis pour les colorer avec RLF, semble vérifiée. Enfin les performances de la méthode "adjacents des adjacents" se situent entre celles de SAT et celles de la paire (SL,LF)

5) La précision des méthodes approximatives décroît pour des graphes de tailles croissantes ; mais croît avec la densité pour des graphes de dimension fixe.
Ceci est logique puisque pour un graphe complet, toutes les méthodes donnent le nombre chromatique.

CHAPITRE III. - LES METHODES EXACTES

III.1 Méthode de N. CRISTOFIDES

III.2 Méthode de Wang

III.3 Méthodes de Brown améliorées

III.4 Méthode de l'énumération implicite

III.5 Résultats de tests

CHAPITRE III : LES METHODES EXACTES

Toutes les méthodes de coloration décrites jusqu'à présent ne sont pas des méthodes exactes, c'est-à-dire elles ne trouvent pas nécessairement le plus petit nombre de couleurs pour une coloration, à savoir le nombre chromatique du graphe. Les méthodes suivantes réalisent cet objectif. Malheureusement, celles-ci ne sont pas implémentables en un temps de calcul polynomial.

(III.1) Méthode de N.CHRISTOFIDES (/11/)

L'outil de base de cette méthode est la recherche des ensembles indépendants maximaux d'un graphe et son principal argument est le théorème du paragraphe (II.1.3.2.)

(III.1.1.) r-sous-graphes maximaux

Un r-sous-graphe du graphe $G(X,E)$ est un sous-graphe $\langle S_r(G)\rangle$, où $S_r(G) \subseteq X$, qui est r-chromatique. S'il n'existe pas d'ensemble de sommets H, $H \supset S_r(G)$ tel que le sous-graphe $\langle H\rangle$ est r-chromatique, alors le graphe $\langle S_r(G)\rangle$ est dit r-sous-graphe maximal de G .

Pour une valeur de r donnée, il y a en général beaucoup de r-sous-graphes maximaux d'un graphe G. Donc, par exemple, un sous-graphe dont les sommets sont formés par un ensemble indépendant maximal de G est un 1-sous-graphe maximal puisque G ne contient pas de sous-graphe avec des sommets supplémentaires qui est 1-chromatique.

Dès lors, le nombre chromatique d'un graphe G est donné par la plus petite valeur r t q. $S_r(G) = X$ pour au moins un r-sous-graphe maximal.

Relation de récurrence

Le théorème énoncé en (II.1.3.2.), peut être maintenant, exploité afin d'établir une relation de récurrence telle que les r-sous-graphes maximaux s'obtiennent à partir des (r-1)-sous-graphes maximaux.
Soit $Q_{r-1}(G)$ la famille des (r-1)-sous-graphes maximaux de G et soit $S_{r-1}^{j}(G)$ l'ensemble des sommets du $j^{ième}$ de ces sous-graphes.

L'ensemble $S_1^{k}(G^j)$ est alors l'ensemble des sommets du $k^{ième}$ 1-sous-graphe maximal (ensemble indépendant maximal) du graphe $G^j = \langle X - S_{r-1}^{j}(G) \rangle$, formé par les sommets de G non contenus dans le sous-graphe $\langle S_{r-1}^{j}(G) \rangle$. La famille $Q_r(G)$ des r-sous-graphes maximaux de G peut être calculée comme suit :
Soit les ensembles $H^i = S_{r-1}^{j}(G) \cup S_1^{k}(G^j)$

$$j = 1,2,\ldots, q_{r-1} \qquad \text{(III.1)}$$
$$k = 1,2,\ldots, q_i^j$$

où q_{r-1} = nombre de r-1-sous-graphes

q_i^j = nombre de 1-sous-graphes

La famille des r-sous-graphes maximaux est contenue dans la famille Θ des ensembles H^i ($i = 1,2, \ldots, q_i^j\ q_{r-1}$) et peut être obtenue par exclusion des ensembles de Θ déjà contenus dans d'autres; donc $Q_r(G)$ est donné par

$$Q_r(G) = \left\{ H^i \mid H^i \in \Theta \text{ et } H^i \not\subseteq H^j \quad \forall\ H^j \in \Theta\ ,\ i \neq j \right\} \qquad \text{(III.2}$$

(III.1.2) Algorithme de coloration basé sur les r-sous-graphes maximaux

Comme on vient de le constater ci-avant, le nombre chromatique d'un graphe G est la plus petite valeur r telle que l'ensemble des sommets $S_r(G)$ d'un r-sous-graphe maximal est l'ensemble entier des sommets du graphe, c'est-à-dire X .

La relation de recurrence (III.1) et(III.2) peut dès lors être utilisée afin de générer progressivement des 1-sous-graphes, 2-sous-graphes, etc..., en vérifiant à chaque étape si l'ensemble X est atteint.
L'algorithme suivant recherche le nombre chromatique d'un graphe G et une coloration réalisant ce nombre.

1) $r = 1$;

chercher $S_1^j(G)$; $j = 1, \ldots, q_r$ les sommets de 1-sous-graphes maximaux (on suppose qu'il existe q_r tels ensembles)

$Q = \{S_1^j(G) \mid j=1 \ldots q_r\}$; Si $q_r = 1$ et $S_1^1(G)=X$, STOP
(le graphe n'a que des sommets isolés)

$j = 1$

2) déterminer un ensemble indépendant maximal $S_1(G^j)$ du graphe
$G^j = \langle X - S_r^j(G) \rangle$

S'il existe aller en 3)

s'ils ont déjà tous été trouvés aller en 6)

3) $S = S_r^j(G) \cup S_1(G^j)$

4) Si S = X, STOP; le nombre (r+1) est le nombre chromatique recherché du graphe; les sous-ensembles de S donnent la coloration cherchée

Si $S \neq X$; aller en 5)

5) (i) Si $S \subseteq S'$ pour $S' \in Q$ aller en 2)
(ii) Si $S \supset S'$ pour $S' \in Q$; $Q=Q-\{S'\}$ pour tels S';
$Q=Q \cup \{S\}$ aller en 2)
(iii) Si ni (i), ni (ii) ne sont vérifiées $Q=Q \cup \{S\}$,
aller en 2)

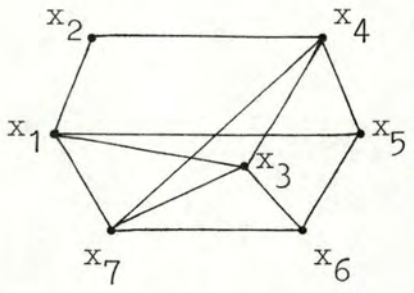
6) . Si $j < q_r$; { $j = j + 1$
aller en 2)

. Si $j = q_r$ { $j = 1$
$r = r + 1$
q_r = nombre d'ensembles de Q
aller en 2

REMARQUE : Si l'algorithme n'est pas arrêté lors du premier S = X (en 4)), alors il continue de produire d'autres r + 1-colorations (si elles existent). Remarquons que l'algorithme ne donnera pas la liste complète des r+1-colorations, mais seulement celles produisant des ensembles indépendants maximaux. Ces colorations peuvent n'être qu'une partie de l'ensemble des r+1-colorations du graphe.

(III.1.3.) EXEMPLE DE COLORATION

Soit le graphe G(X,E), n = 7 suivant



Etudions le comportement de l'algorithme sur ce graphe (dont $\chi(G) = 3$)

1) Recherche des 1-sous-graphes maximaux

$S_1^1(G) = \{1,4,6\}$; $S_1^2(G) = \{2,3,5\}$; $S_1^3(G) = \{2,5,7\}$;

$S_1^4(G) = \{2,6\}$;

$q_1 = 4$; $Q = \{ S_1^1(G) , S_1^2(G) , S_1^3(G) , S_1^4(G) \}$

On applique, dans la suite, les étapes 2)- 5) pour ces 1-sous-graphes :

- $S_1^1(G)$: (j = 1)

2) $G^1 = \langle X - S_1^1(G) \rangle = \langle \{2,3,5,7\} \rangle$; $S_1(G^1) = \{ 2,3,5 \}$

3) $S = \{ 1,4,6 \uparrow 2,3,5 \}$

5) $S \supset S_1^2(G)$; $S \supset S_1^4(G) \Rightarrow Q = \{ \{ 1,4,6 \uparrow 2,3,5 \} , \{2,5,7\}$

2) $S_1(G^1) = \{2,5,7\}$

3) $S = \{1,4,6 \uparrow 2,5,7\}$

5) $Q = \{\{1,4,6 \uparrow 2,3,5\}, \{1,4,6 \uparrow 2,5,7\}\}$

- $S_1^2(G)$; $(j = 2)$

2) $G^2 = \langle X\text{-}S_1^2(G) \rangle = \langle \{1,4,6,7\} \rangle$; $S_1(G^2) = \{1,4,6\}$

3) $S = \{2,3,5 \uparrow 1,4,6\}$; éliminé par 5)(i) ;

2) $S_1(G^2) = \{7\}$

3) $S = \{2,3,5 \uparrow 7\}$

5) $Q = \{\{1,4,6 \uparrow 2,3,5\}, \{1,4,6 \uparrow 2,5,7\}, \{2,3,5 \uparrow 7\}\}$ (*)

- $S_1^3(G)$; $j = 3$

2) $G^3 = \langle \{1,3,4,6\} \rangle$; $S_1(G^3) = \{1,4,6\}$

3) $S = \{2,5,7 \uparrow 1,4,6\}$; éliminé par 5) (i)

2) $S_1(G^3) = \{3\}$

3) $S = \{2,5,7 \uparrow 3\}$; éliminé par 5) (i)

- $S_1^4(G)$; $j = 4$

2) $G^4 = \langle \{1,3,4,5,7\} \rangle$; $S_1(G^4) = \{1,4\}$

3) $S = \{2,6 \uparrow 1,4\}$; éliminé par 5) (i)

2) $S_1(G^4) = \{3,5\}$

3) $S = \{2,6 \uparrow 3,5\}$; éliminé par 5) (i)

2) $S_1(G^4) = \{5,7\}$

3) $S = \{2,6 \uparrow 5,7\}$; éliminé par 5) (i)

6) $\Rightarrow$ $j = 1$; $r = 2$, $q_r = 3$

Donc, à la fin de la première itération 2) - 5), la famille Q des ensembles $S_2^j(G)$ (2-sous-graphes maximaux) a été trouvée (cfr. *)
Remarquons que 5 des 8 ensembles générés S ont été éliminés en 5) .
Donc $Q = \{ \{1,4,6 \uparrow 2,3,5\}, \{1,4,6 \uparrow 2,5,7\}, \{2,3,5 \uparrow 7\} \}$

- $S_2^1(G)$; $j = 1$

2) $G^1 = \langle X - S_2^1(G) \rangle = \langle \{7\} \rangle$; $S_1(G^1) = \{7\}$

3) $S = \{1,4,6 \uparrow 2,3,5 \uparrow 7\}$

4) $S = X \Rightarrow$ STOP : $r = r+1 = 3$ est le nombre chromatique

La coloration optimale sera $\{1,4,6\}, \{2,3,5\}, \{7\}$;

Si on avait continué l'algorithme, une autre coloration optimale obtenue aurait été $\{1,4,6\}, \{2,5,7\}, \{3\}$.
Remarquons que les colorations possibles telles que $\{3,5\}$, $\{1,4,6\}$, $\{2,7\}$ ne sont pas produites par l'algorithme puisque leurs ensembles ne sont pas indépendants maximaux.

(III.1.4) RECHERCHE D'ENSEMBLES MAXIMAUX INDEPENDANTS (/12/)

Puisqu'il est évident qu'un ensemble de noeuds C est une clique d'un graphe G si et seulement si C est un ensemble indépendant du graphe complément $\overline{G}$ de G ; nous allons proposer un algorithme pour construire les cliques maximales d'un graphe. Le problème initial de la recherche d'ensembles indépendants maximaux peut alors se résoudre de la manière suivante :

(i) construire $\overline{G}$ le graphe complément de G
(ii) appliquer à $\overline{G}$ l'algorithme pour trouver les cliques maximales

Il est possible de trouver toutes les cliques maximales d'un graphe à l'aide d'une méthode de recherche dans un arbre. Pour présenter cette méthode nous allons considérer le problème plus général suivant :

Soient, donnés, un graphe G (X,E), un sous-graphe $\tilde{G}=(\tilde{X},\tilde{E})$ de G, et un ensemble de noeuds $N \subseteq X-\tilde{X}$; il faut alors trouver toutes les cliques maximales de $\tilde{G}$ qui ne sont pas contenues dans l'ensemble $\Gamma(x_i)$ des adjacents dans G de n'importe quel noeud x_i de N

Remarquons que notre problème original de trouver toutes les cliques maximales d'un graphe G est un cas particulier de ce nouveau problème dans lequel $\tilde{G} = G$ (ce qui implique $N = \emptyset$).

Pour résoudre le problème général, notons par $\tilde{S}$ l'ensemble de toutes les cliques maximales de $\tilde{G}$ et par S l'ensemble des cliques recherchées, nous pouvons alors écrire :

$$S = \left\{ C \in \tilde{S} \mid C \not\subseteq \Gamma(x_i) \ , \forall \ x_i \in N \right\} \quad \text{(III.3)}$$

Maintenant, cet ensemble S peut être construit grâce au processus suivant :

1) Nous déterminons d'abord, si l'ensemble de noeuds $\tilde{X}$ de $\tilde{G}$ satisfait la condition :

$$\tilde{X} \subseteq \Gamma(x_i) \text{ pour un } x_i \in N. \qquad \text{(III.4)}$$

Si cette condition est vérifiée, alors pour toute clique $C \in \tilde{S}$ nous avons :

$$C \subseteq \Gamma(x_i) \text{ pour un } x_i \in N \qquad \text{(III.5)}$$

qui implique (par (III.3))que l'ensemble recherché S est vide et le problème est résolu; sinon nous exécutons l'étape 2)

2) Si le sous-graphe $\tilde{G}$ $(\tilde{X},\tilde{E})$ est complet alors $\tilde{S} = \{\tilde{X}\}$, et puisque la condition (III.4) n'est pas vérifiée, il suit de (III.3) que $S = \{\tilde{X}\}$, ainsi le problème est résolu; sinon nous exécutons l'étape 3)

3) Puisque le sous-graphe $\tilde{G}$ n'est pas complet, il possède au moins un noeud x_k qui n'est pas adjacent à tous les noeuds de $\tilde{X}$.
Exprimons, maintenant, l'ensemble désiré S comme l'union de deux ensembles disjoints

$$S = S_k \cup S_{\bar{k}} \qquad \text{(III.6.a)}$$

où

$$S_k = \{ C \in S \mid x_k \in C \} \quad S_{\bar{k}} = \{ C \in S \mid x_k \notin C \} \qquad \text{(III.6.b)}$$

De (III.3) et(III.6.b) on déduit que

$$S_k = \{ C \in \tilde{S}_k \mid C \nsubseteq \Gamma(x_i), \forall\, x_i \in N_k \} \qquad \text{(III.7)}$$

où $\tilde{S}_k$ est l'ensemble de toutes les cliques maximales du sous-graphe $\tilde{G}_k$ de $\tilde{G}$ qui contient seulement le noeud x_k et ses adjacents, et $N_k = N$, tandis que

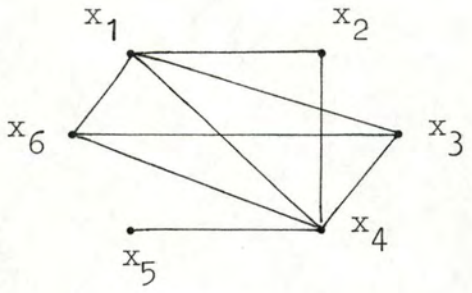
$$S_{\bar{k}} = \{ C \in \tilde{S}_{\bar{k}} \mid C \nsubseteq \Gamma(x_i) \,,\, \forall\, x_i \in N_{\bar{k}} \} \qquad \text{(III.8)}$$

où $\tilde{S}_{\bar{k}}$ est l'ensemble de toutes les cliques maximales sur le sous-graphe $\tilde{G}_{\bar{k}}$ de $\tilde{G}$ obtenu en enlevant x_k et $N_{\bar{k}} = N \cup \{x_k\}$. Donc le problème de trouver l'ensemble S de (III.3) est réduit à deux sous-problèmes, comprenant la détermination séparée des ensembles S_k et $S_{\bar{k}}$ définis par (III.7) et (III.8) respectivement.

On observera que lesproblèmesde construire les ensembles $S_{\bar{k}}$ et S_k sont, tous les deux, plus simplesque le problème initial de trouver S ; car les sous-graphes $\tilde{G}_k$ et $\tilde{G}_{\bar{k}}$ contiennent moins de sommets que $\tilde{G}$. Par conséquent, l'application répétée du processus décrit ci-dessus produit des sous-problèmes qui sont "triviaux" (en ce sens que pour chaque sous-problème, soit la condition (III.4) est vérifiée, soit le sous-graphe $\tilde{G}$ est complet).

Exemple de calcul

Soit le graphe G(X,E) dont on doit trouver les cliques maximales



L'application de la méthode est représentée par la recherche arborescente de la figure (III.1)

Le nombre d'appels récursifs du processus peut souvent être réduit considérablement en appliquant le théorème suivant :

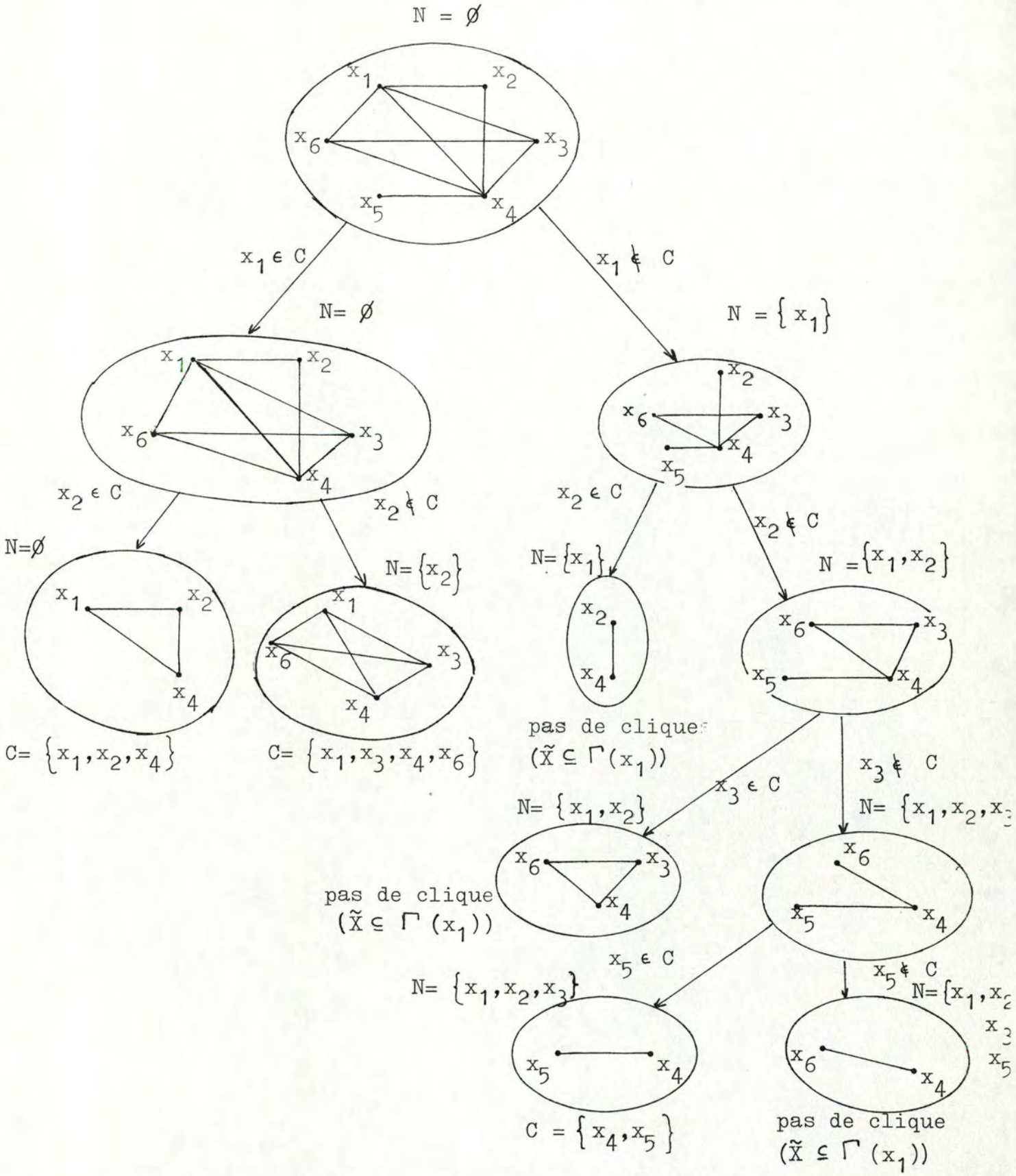
Théorème 1

Soient G(X,E) un graphe, C une clique maximale de G, et x_k un noeud tel que $x_k \notin C$;

Si x_k possède un adjacent x_1 vérifiant

$$\{x_1\} \cup \Gamma(x_1) \subseteq \{x_k\} \cup \Gamma(x_k) \qquad \text{(III.9)}$$

alors $x_1 \notin C$


N = ∅
x1 ∈ C
x1 ∉ C
N= ∅
N = {x1}
x2 ∈ C
x2 ∉ C
x2 ∈ C
x2 ∉ C
N=∅
N= {x2}
N= {x1}
N = {x1, x2}
C= {x1, x2, x4}
C= {x1, x3, x4, x6}
pas de clique
(X̃ ⊆ Γ(x1))
x3 ∈ C
x3 ∉ C
N= {x1, x2}
N= {x1, x2, x3
pas de clique
(X̃ ⊆ Γ(x1))
x5 ∈ C
x5 ∉ C
N= {x1, x2, x3}
N= {x1, x2
x3
x5
C = {x4, x5}
pas de clique
(X̃ ⊆ Γ(x1))

FIGURE(III.1)

Preuve

Puisque la clique C est maximale, et que $x_k \notin C$, il est évident que la clique C contient au moins un noeud x_i qui n'est pas adjacent à x_k ; mais alors x_i n'est pas adjacent à n'importe quel noeud x_l pour lequel la condition (III.9) est vérifiée, et dès lors C ne peut contenir aucun de ces noeuds.

Ce théorème peut être utilisé dans la méthode décrite ci-dessus: quand nous construisons les sous-graphes $\tilde{G}_k$ et $\tilde{G}_{\overline{k}}$ de $\tilde{G}$ nous pouvons omettre de $\tilde{G}_{\overline{k}}$ n'importe quel noeud x_l tel que la condition (III.9) est vérifiée sur $\tilde{G}$.
En illustration, dans la figure (III.1) au moment où l'on construit $\tilde{G}_1$ et $\tilde{G}_{\overline{1}}$, nous pouvons omettre de $\tilde{G}_{\overline{1}}$ les noeuds x_2, x_3, et x_6 ; la procédure modifiée, dans laquelle nous avons utilisé cette simplification, est représentée par la figure (III.2).

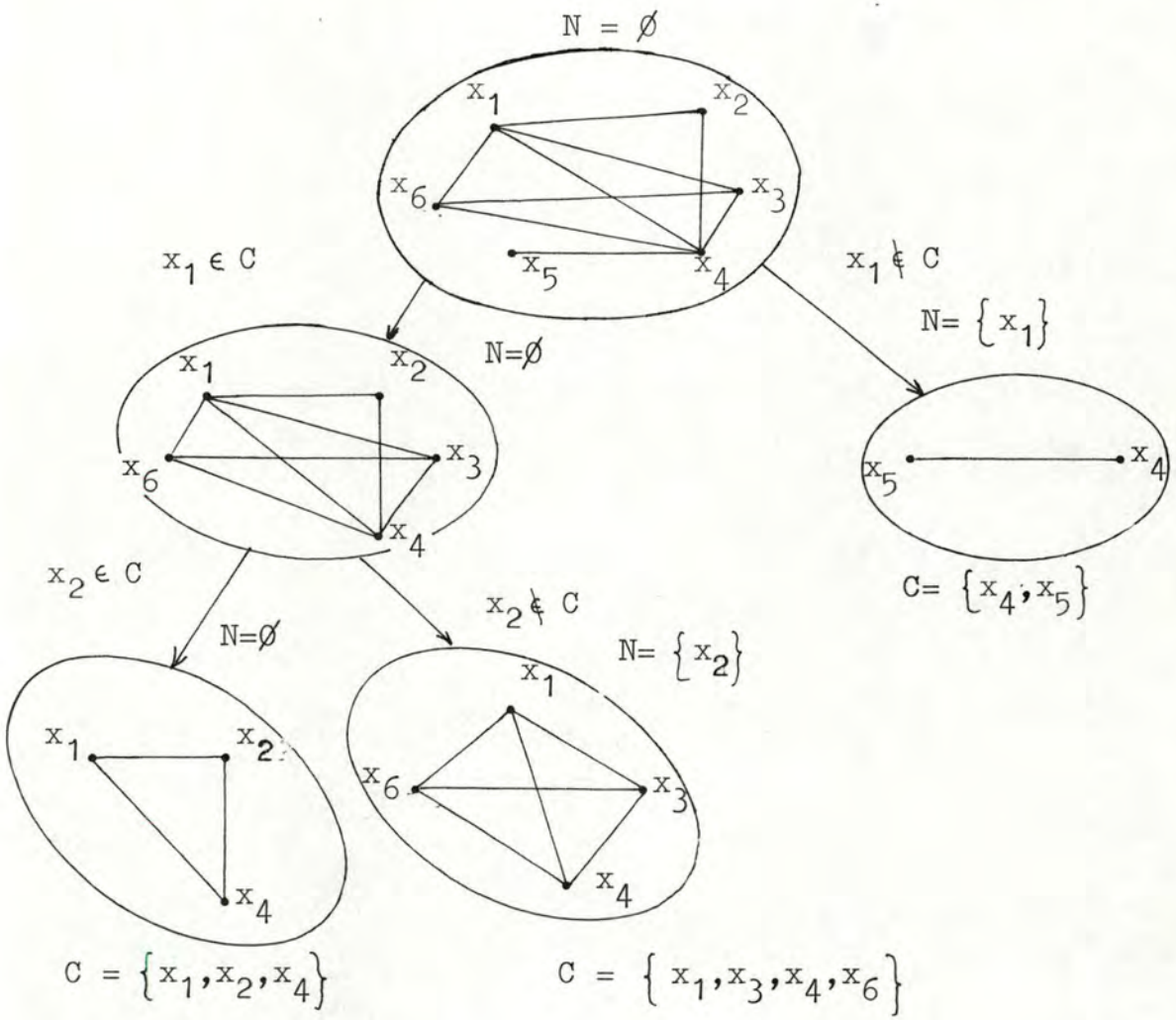


FIGURE (III.2)

Analyse de complexité

Puisque l'itération de l'algorithme présenté ci-dessus requiert $\mathcal{O}(n^2)$ temps de calcul; il nous suffit d'estimer le nombre de fois que celle-ci sera exécutée pour connaître la complexité de la recherche des cliques maximales. Pour cela, utilisons la représentation sous forme d'arbre de notre recherche (cfr. figures (III.1) et (III.2)).

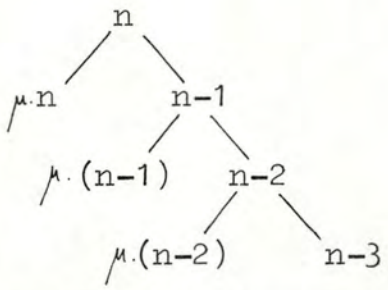
Chaque feuille de l'arbre représente un graphe $\widetilde{G}$ traité, les feuilles de gauche et de droite suivantes sont obtenues, l'une en considérant le sous-graphe de $\widetilde{G}$ engendré par x_k et ses adjacents, l'autre en éliminant ce même x_k de $\widetilde{G}$. Donc plus on descend dans l'arbre, moins les graphes $\widetilde{G}$ ont d'éléments et au pire, notre recherche s'arrêtera au moment où les feuilles terminales seront réduites à un seul noeud.

Si la racine de l'arbre contient n éléments et que le passage d'un niveau i à un niveau i+1 entraîne la diminution de 1 et 1 seul élément du prédécesseur, alors, on s'arrêtera après n-1 niveaux et, puisque l'arbre est binaire le nombre de feuilles sera

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1 \qquad \text{(III.10)}$$

Dans notre arbre, si le passage d'une feuille vers sa suivante de droite, s'obtient en retirant un élément de la feuille "mère", la feuille de gauche suivante garde en général beaucoup moins d'éléments. Ainsi, en supposant que la racine comporte n noeuds la feuille gauche du niveau 1 possèdera $\mu \cdot n$ éléments, où $\mu = d/n$ est la densité du graphe, et la feuille droite aura (n-1) éléments. La formule (III.10) nous apprend alors que le sous-arbre de gauche portera au plus $2^{\mu n} - 1$ feuilles tandis que le sous-arbre de droite dont la racine est garnie de (n-1) noeuds peut être décomposé en un sous-arbre de gauche

avec une racine à $\mu.(n-1)$ éléments et qui admettra au plus $2^{\mu.(n-1)}-1$ feuilles, et un sous-arbre de droite avec une racine à n - 2 éléments qui peut également faire l'objet d'une décomposition, etc, ...



On descend comme cela dans l'arbre tant que $\mu.(n-i) \geq 2$, $i = 1,2,3,\ldots$. On s'arrêtera donc quand :

$$i = n - E\left[\frac{2}{\mu}\right] \qquad \text{(où } E\,[x]\text{, désigne la partie entière de x)}$$

A ce niveau le processus de décomposition évoqué plus haut donnera un sous-arbre de gauche dont la racine regroupe 2 noeuds, et le sous-arbre de droite dont la racine comporte $E\left[\frac{2}{\mu}\right] - 1$ sommets isolés.

Puisque la recherche de cliques maximales dans un graphe comprenant n sommets isolés génère un arbre binaire garni de (2n-1) feuilles, la méthode appliquée à un graphe quelconque engendrera un arbre de

$$\sum_{i=0}^{n-E\left[\frac{2}{\mu}\right]} (2^{\mu(n-i)}-1) + 2\left(E\left[\frac{2}{\mu}\right] - 1\right) - 1 + \left(n - E\left[\frac{2}{\mu}\right]\right)$$

$$= \sum_{i=0}^{n-E\left[\frac{2}{\mu}\right]} (2^{\mu(n-i)}) - n + E\left[\frac{2}{\mu}\right] - 1 + 2\,E\left[\frac{2}{\mu}\right] - 3 + n - E\left[\frac{2}{\mu}\right]$$

$$\simeq 2^{\mu(n+1)} + 2\,E\left[\frac{2}{\mu}\right] - 8$$

feuilles.

Or, à chaque feuille est associé un traitement exigeant $\delta(n^2)$ temps de calcul, donc la complexité de l'algorithme analysé est $\delta(n^2(2^{\mu n} + 2\,E\left[\frac{2}{\mu}\right])$

(III.1.5.) ETUDE DE COMPLEXITE DE LA METHODE DE CHRISTOFIDES

Pour cela, remarquons que, par l'intermédiaire de l'étape 2), le processus de calcul des ensembles indépendants maximaux est exécuté pourchaque $S_r^j(G)$.
Comme il a été calculé ci-dessus, ce processus consomme $\delta(n^2(2^{\mu n}+\left|\frac{2}{\mu}\right|))$ temps de calcul. D'autre part, puisqu'il n'existe pas plus de $3^{(n/3)}$ ensembles indépendants maximaux dans un graphe de n sommets, le nombre de $S_r^j(G)$ ne peut excéder :

$$\sum_{r=1}^{n/3} 3^{nr/3} = \frac{3^{n/3} - 3^{(n/3)^2}\cdot 3^{n/3}}{1 - 3^{n/3}} \simeq 3^{(n/3)^2}$$

Donc, l'ensemble des exécutions de l'étape 2) ne dépassera pas $\delta(n^2(2^{\mu n}+\left|\frac{2}{\mu}\right|).3^{(n/3)^2})$ temps de calcul.

L'étape 5) quant à elle, pour n fixé, nécessite

$$n.\sum_{i=0}^{q_r-1} i = \frac{q_r\,(q_r-1)}{2}$$. n comparaisons (où q_r est le nombre de $S_r^j(G)$, $j = 1, \ldots, q_r$).

Or, ce nombre q_r est majoré par $3^{nr/3}$. De ce fait, l'ensemble des étapes 5) coûte

$$n.\sum_{r=1}^{n/3} \frac{3^{nr/3}\,(3^{nr/3}-1)}{2} = \frac{n}{2}\left(\sum_{r=1}^{n/3} 3^{2nr/3} - \sum_{r=1}^{n/3} 3^{nr/3}\right)$$

$$= \frac{n}{2}\left[\left(\frac{3^{2(n/3)^2}\cdot 3^{2n/3} - 3^{2n/3}}{3^{2n/3}-1}\right) - 3\frac{(n/3)^2}{3^{n/3}-1}\cdot\frac{3^{n/3}-3^{n/3}}{}\right]$$

$$\simeq \frac{n}{2}\left[3^{2(n/3)^2} - 3^{(n/3)^2}\right] \quad \text{comparaisons}$$

Finalement, puisque les étapes 1), 3), 4), et 6), consomment un temps de calcul négligeable, la complexité de la méthode de N. CHRISTOFIDES est de l'ordre de

$$\mathcal{O}\left((n^2(2^{\mu \cdot n_{+\left|\frac{2}{\mu}\right|}} - n)3^{(n/3)^2} + n\, 3^{2(n/3)^2} \right)$$

(III.2) METHODE DE WANG (/13/)

Nous allons maintenant améliorer l'algorithme de coloration par r-sous-graphes maximaux en diminuant le nombre total de sous-ensembles maximaux à analyser et d'appeler l'algorithme de calcul des ensembles maximaux indépendants une seule fois (les ensembles maximaux indépendants de $\langle X-S \rangle$ se déduisent facilement des ensembles maximaux de G)

(III.2.1) REDUCTION DU NOMBRE D'ENSEMBLES MAXIMAUX INDEPENDANTS A ANALYSER

(III.2.1.1.) Quelques résultats théoriques

Lemme 1 Soit $x \in X$ quelconque et $S^1, S^2, \ldots, S^k$ les ensembles maximaux indépendants de G contenant x; alors il existe 1 χ-coloration de G telle que les sommets colorés de même que x sont donnés par S^i $1 \leqslant i \leqslant k$

Preuve Soit n le nombre chromatique de G et $\{X_1, X_2, \ldots, X_n\}$ une coloration de G . Sans perte de généralité, $x \in X_1$. Si $X_1 = S^i$, $1 \leqslant i \leqslant k$, le lemme est prouvé. Sinon, il existe i, $1 \leqslant i \leqslant k$ tel que $X_1 \subset S^i$
Soient X_2' , X_3' , ..., X_n' les ensembles obtenus en enlevant des sommets de $S^i - X_1$ des ensembles $X_2, \ldots, X_n$.
Alors la partition $\{S^i, X_2', X_3', \ldots, X_n'\}$ est une χ-coloration G.

Lemme 2 Soit S un ensemble maximal indépendant de G, alors le nombre chromatique de $\langle X-S \rangle$ vaut $\chi(G)-1$ ssi S appartient à la partition d'une coloration chromatique de G.

Preuve : suit immédiatement des définitions

Théorème 2 : Soit $x \in X$ et $S^1, S^2, \ldots, S^k$ les ensembles maximaux indépendants contenant x alors $\chi(G) = 1 + \min_{1 \leq i \leq k} \{\chi(\langle X - S^i \rangle)\}$

Preuve Soit $n = \chi(G)$; alors $\chi(\langle X - S^i \rangle) \geq n-1$. Par le lemme 1, il existe i, $1 \leq i \leq k$ tel que S^i fait partie d'une χ-coloration de G. Le lemme 2 entraîne que $\chi(\langle X - S^i \rangle) = n-1$, d'où le résultat.

(III.2.1.2) UN ARBRE REDUIT DE SOUS-GRAPHES

Le processus de coloration de G présenté au paragraphe (III.1), peut être considéré comme une recherche de chemin minimal dans un arbre de sous-graphes de G.

Au lieu de considérer l'arbre total des sous-graphes de G , on ne traitera qu'un arbre réduit qui sera construit de la façon suivante :

- la racine de l'arbre est G (niveau o)
- chercher un sommet x contenu dans le plus petit nombre d'ensembles maximaux indépendants de G. Soient $S^1, \ldots, S^k$ ces ensembles.
- les sous-graphes $\langle X - S^1 \rangle$, $\langle X - S^2 \rangle$,..., $\langle X - S^k \rangle$ satisfont le théorème 2 pour déterminer $\chi(G)$; ces sous-graphes induits forment le premier niveau de l'arbre réduit (rappelons que pour l'algorithme du paragraphe précédent, ce niveau est formé par les sous-graphes obtenus en déduisant tous les ensembles maximaux indépendants de G)
- pour chaque sous-graphe de niveau 1, $G' = \langle X - S^i \rangle$, $1 \leq i \leq k$, soit $y \in G'$ le sommet contenu dans le plus petit nombre d'ensembles maximaux indépendants du G'. Soient $T^1, T^2, \ldots T^l$ ces ensembles, alors les sous-graphes $\langle X - (S^i \cup T^1) \rangle$, ... $\langle X - (S^i \cup T^l) \rangle$ satisfont le théorème 2 et forment le niveau 2 de l'arbre, ...

La méthode de CHRISTOFIDES utilise tous les ensembles indépendants maximaux de G et se base donc sur un arbre maximal de sous-graphesde G contenant l'arbre réduit décrit ci-dessus.

Le chemin de la racine de l'arbre vers un noeud terminal (sous-graphe vide) est une coloration de G; la longueur du chemin est égale à $\chi(G)$.
Le nombre d'itérations exigé pour calculer $\chi(G)$ est donc proportionnel au nombre total de chemins de l'arbre.

Comme l'arbre maximal contient l'arbre réduit, l'algorithme de CHRISTOFIDES sera amélioré (en ce qui concerne le nombre d'itérations) lorsque l'on ne considère que l'arbre réduit.

(III.2.1.3) ANALYSE QUANTITATIVE

Etudions la taille de l'arbre maximal et de l'arbre réduit pour deux classes spéciales de graphes.

* Soit un graphe G formé par m copies disjointes de sous-graphes complets de p/m sommets. G possède donc p sommets. On trouve un ensemble maximal indépendant typique de G en prenant un sommet de chacun des sous-graphes complets de G. D'où, il existe $(p/m)^m$ ensembles maximaux indépendants de G.

  Soit G' un sous-graphe de niveau i de G. Ce G' sera formé par m copies disjointes du graphe complet avec (p/m-i) sommets. Le nombre d'ensembles maximaux indépendants de G' est $(p/m-i)^m$. Le nombre de noeuds dans l'arbre sortant de G' est dès lors $(p/m-i)^m$. En plus, un sommet fixé de G' est contenu dans exactement $(p/m-i)^{m-1}$ ensembles maximaux indépendants de G' ce qui entraîne qu'il y a donc $(p/m-i)^{m-1}$ branchemen dans le sous-arbre réduit. Tous les chemins du sous-arbre de la racine aux noeuds terminaux sont de même longueur. Le nombre de chemins les plus courts est égal au nombre de noeuds terminaux du sous-arbre de G.

Il y a $((p/m)!)^m$ noeuds terminaux dans l'arbre maximal de G et $((p/m)!)^{m-1}$ noeuds terminaux dans l'arbre réduit de G. La méthode de "l'arbre réduit" diminue donc le nombre d'itérationsde (p/m)! fois.

* Soit le graphe G_{pq} de p sommets et p(p-1)/2 arêtes dont chacune existe avec un probabilité q. Matula montrait que le nombre de cliques de ce graphe est

$$\sum_{d=1}^{p} \binom{p}{d} \; q^{d(d-1)/2} \; (1 - q^d)^{p-d} .$$

Le nombre de noeuds de niveau 1 de l'arbre maximal de G_{pq} est égal au nombre d'ensembles indépendants maximaux que l'on peut obtenir facilement en remplaçant 1-q par q dans l'expression précédente (vu la complémentarité des notions de clique et d'ensemble indépendant). Ce nombre vaut :

$$E_2 = \sum_{d=1}^{p} \binom{p}{d} \; (1-q)^{d(d-1)/2} \; (1- (1 -q)^d)^{p-d}$$

Le nombre de noeuds de niveau 1 de l'arbre réduit est généralement plus petit que le nombre d'ensembles maximaux indépendants contenant un point fixé de G_{pq} qui vaut :

$$E_1 = \sum_{d=1}^{p} \binom{p-1}{d-1} \; (1-q)^{d(d-1)/2} (1-(1-q)^d)^{p-d}$$

La borne E_1 est bien inférieure à E_2 en général.

(III.2.1.4) <u>L'ALGORITHME</u> (ALGO 1)

On utilise l'algorithme de CHRISTOFIDES en changeant légèrement le critère de choix des ensembles maximaux indépendants.

1) $r = 1$

Chercher $S_1^j(G)$, $j=1,\ldots,q_r$ les 1-sous-graphes maximaux de G, soit x_p = le sommet contenu dans le plus petit nombre de $S_1^j(G)$. $\quad Q = \left\{ S_r^j(G) \mid S_r^j(G) \ni x_p, j = 1 \ldots q_r \right\}$

Si $q_r = 1$ et $S_1^1(G) = X$, STOP

$j = 1$

2) Soit x_p le sommet contenu dans le plus petit nombre de $S_1(G^j)$, ensembles maximaux indépendants de $G^j = \langle X - S_r^j(G) \rangle$

On ne considère que $S_1(G^j) \ni x_p$; s'il en existe aller en 3)

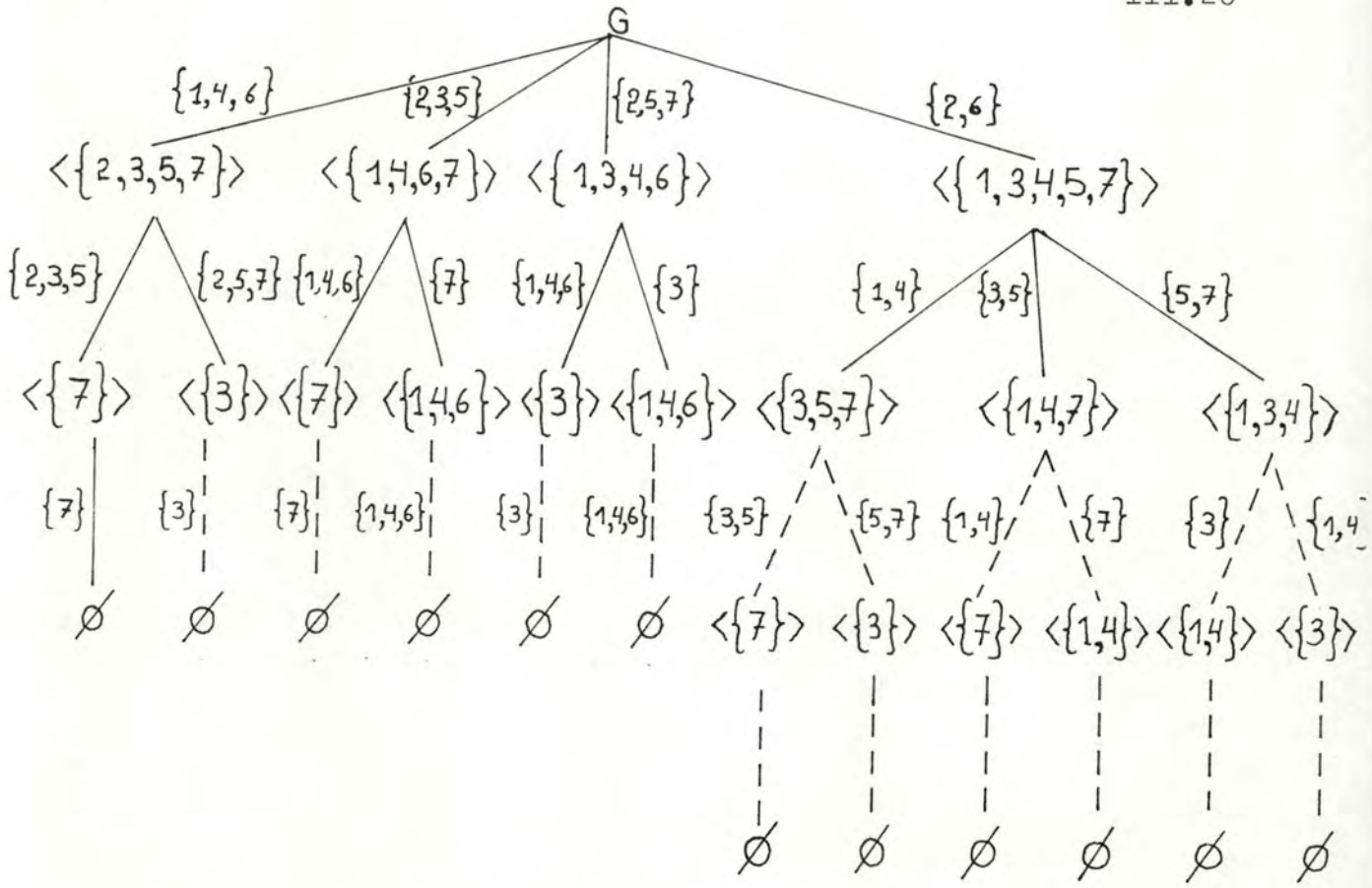
S'ils ont déjà été tous traités aller en 6)

3)
.
.
.
6) } idem Christofides

(III 2.1.5.) EXEMPLE

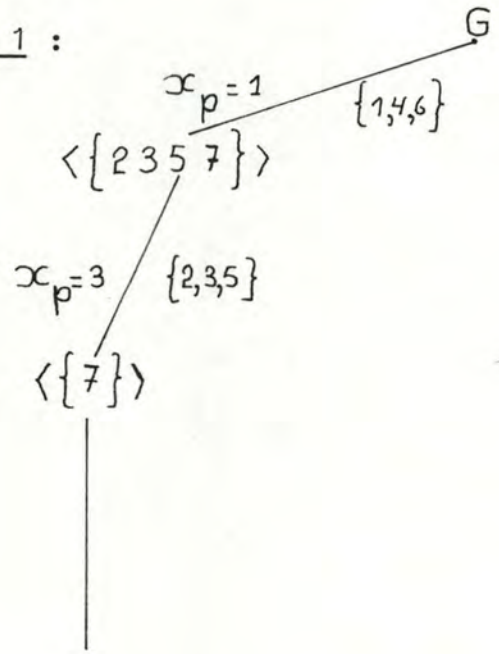
Comparons l'algorithme de CHRISTOFIDES et ALGO 1 grâce aux arbres des ensembles maxi. indépendants traités sur l'exemple étudié en (III.1.3.)

1) Christofides



Le chemin —— marque la partie de l'arbre parcourue par l'algorithme de Christofides

2) <u>ALGO 1</u> :



(III.2.2.) APPEL à L'ALGORITHME DE CALCUL DES ENSEMBLES MAXIMAUX INDEPENDANTS UNE SEULE FOIS

L'appel à l'algorithme pour rechercher les ensembles maximaux indépendants n'est absolument nécessaire qu'à l'itération 1) de ALGO 1: en effet à l'itération 1), on détermine tous les ensembles maximaux indépendants de G; ces ensembles peuvent être utilisés pour déterminer les ensembles maximaux indépendants du graphe réduit à l'itération 2) : de façon à pouvoir supprimer tout appel supplémentaire à l'algorithme décrit en (III.1.4)

Soient $S_1^i(G)$ les ensembles maximaux indépendants de G, i = 1, ... k, alors les ensembles maximaux de $G^j = \langle X - S_1^j(G) \rangle$ (j= 1,...k) se déterminent comme suit :

* $S_1'^i = S_1^i(G) - S_1^j(G)$ $\quad i = 1, \ldots, k$
* éliminer les $S_1'^i$ tels que $\exists$ l t q $\quad S_1'^l \supseteq S_1'^i$
* Les $S_1'^i$ restants forment les ensembles maximaux indépendants de G.

Exemple : soit l'exemple (III.1.3)

$$S_1^i(G) = \{\{1,4,6\}, \{2,3,5\}, \{2,5,7\}, \{2,6\}\} \quad i=1,\ldots,4$$
$$S_1^{1''}(G) \quad S_1^{2''}(G) \quad S_1^{3''}(G) \quad S_1^{4''}(G)$$

$$G^2 = \langle \{1,4,6,7\} \rangle = \langle X - S_1^2(G) \rangle$$

$$* \; S_1'^1 = \{1,4,6\} / \{2,3,5\} = \{1,4,6\}$$

$$S_1'^2 = \phi \; ; \; S_1'^3 = \{2,5,7\} / \{2,3,5\} = \{7\}$$

$$S_1'^4 = \{2,6\} / \{2,3,5\} = \{6\}$$

* éliminer $S_1'^2$, $S_1'^4$
* les ensembles maximaux indépendants sont

$$\{1,4,6\} \; , \{7\}$$

(III.2.3) UTILISATION DE LA DEPTH-FIRST SEARCH

On vient de présenter le processus de coloration d'un graphe comme une recherche d'un chemin minimal dans un arbre de sous-graphes de G. Jusqu'à présent, cette recherche a été réalisée par une breadth-first-search, l'algorithme que nous vous présenterons ci-dessous résoud ce problème grâce à une depth-first search (cfr.§ (II.1.1.5) et § (II.1.1.6)).

Puisque tous les noeuds des niveaux, qui sont inférieurs à la longueur du chemin le plus court, doivent être visités par les deux méthodes, le nombre de noeuds examinés lors de la depth-first search est en général plus grand que lors de la breadth-first search. Toutefois, la capacité de stockage requise par la depth-first search est la plus faible des deux. Quand les noeuds du $i^{ième}$ niveau sont explorés, tous les noeuds des niveaux o à i-1 doivent être mémorisés lors d'une breadth-first search, tandis qu'avec la depth-first search, les noeuds du $j^{ième}$ niveau, $1 \leqslant j \leqslant i$, devant être stockés sont seulement ceux branchés à un noeud commun du niveau (j-1).

(III.2.3.1) L'ALGORITHME DE WANG

Soient G(X,E) : le graphe à colorer,

- T : l'ensemble de sommets de G qui ne sont pas présents dans le sous-graphe courant au niveau r
- b_n : l'ensemble indépendant maximal de $\langle X-T \rangle$ traité au niveau n
- e_n : le nombre d'ensembles indépendants maximaux de $\langle X-T \rangle$ du niveau n qui n'ont pas encore été traités
- g : sera le nombre chromatique du graphe.

On utilise aussi une pile pour stocker tous les ensembles maximaux indépendants des niveaux 1 à n qui n'ont pas encore été traités.

1) $T=\emptyset$; pile=$\emptyset$; $n=o$; $b_o=\emptyset$; $g=\#X$; calculer les ensembles indépendants maximaux de X : $M_1,\ldots M_m$

2) Si $n \geq g$ alors $T=T-b_n$; aller en 6)
sinon aller en 3)

3) Si $X-T=\emptyset$ alors $g=n$; $T=T-b_n$; colorer b_i avec i, $i=1,2,\ldots g$;
aller en 6)
sinon continuer en 4)

4) Déduire les ensembles indépendants maximaux de $\langle X-T \rangle$ de $M_1-T, M_2-T, \ldots, M_m-T$, et notons les $S_1,\ldots,S_t$.

5) Choisir $u \in X-T$ tel que u est contenu dans le plus petit nombre d'ensembles maximaux indépendants de $\langle X-T \rangle$.
Notons les $S_{u_1} \ldots S_{u_r}$.
Mettre S_{u_i} $(i=1,2,\ldots,r)$ dans la pile;
$n = n+1$; $e_n = r$;

6) Si $n=o$ alors STOP
sinon continuer en 7)

7) Si $e_n=o$ alors $n=n-1$; $T=T-b_n$ et aller en 6)
sinon continuer en 8)

8) Mettre le sommet de la pile dans b_n; $e_n=e_n-1$, $T=TUb_n$,
aller en 2)

(III.2.3.2) <u>ETUDE DE LA COMPLEXITE DE L'ALGORITHME DE WANG</u>

Comme on l'a vu précédemment, l'étape 1) est exécutée une seule fois et requiert $\delta(n^2.(2^{(\mu.n)}+|\frac{2}{\mu}|))$ temps de calcul.

En utilisant le fait que le nombre d'ensembles indépendants maximaux d'un graphe comportant n sommets n'excède pas $3^{(n/3)}$, on déduit que l'itération 4) requiert au plus

$3^{(n/3)}$. n éliminations de sommets et

$3^{\frac{(n/3)(3^{n/3}-1)}{2}}$. n comparaisons

Le temps de calcul des autres étapes étant négligeable , il nous reste à connaître le nombre de fois que l'étape 4) sera exécutée. Puisqu'il ne peut y avoir plus de $3^{(n(1-\mu)-1)/3}$ ensembles indépendants maximaux comprenant un même sommet d'un graphe à n noeuds, l'itération 4) sera exécutée:

$$\sum_{r=1}^{n/3} 3^{r(n(1-\mu)-1)/3} = \frac{3^{n(n(1-\mu)-1)/9} \cdot 3^{(n(1-\mu)-1)/3} - 3^{(n(1-\mu)-1)/3}}{3^{(n(1-\mu)-1)/3} - 1}$$

$$\simeq 3^{n(n(1-\mu)-1)/9}$$

$$\simeq 3^{n^2 \cdot (1-\mu)/9} \text{ fois}$$

Donc, le temps de calcul requis par l'exécution des différentes itérations 4) s'élève à $\delta(3^{n^2.(1-\mu)/9}.n.\ (3^{n.2/3} + 3^{n/3}))$

Finalement, la complexité de la méthode de WANG est de l'ordre de

$$\delta((n^2.(2^{\mu.n}+\lfloor \frac{2}{\mu} \rfloor)) + 3^{n^2(1-\mu)/9}.n)$$

(III.3) METHODESDE BROWN améliorées (/8/,/14/)

RANDALL-BROWN a développé un algorithme qui évite les redondances dans l'énumération des solutions du problème de la coloration d'un graphe. Dans ce but, il propose de construire l'arbre suivant :

Définition : Une solution partielle de niveau p est n'importe quelle affectation de couleur dans laquelle seuls les noeuds $x_1, \ldots, x_p$ ont été colorés.

Construction de l'arbre

Soit K_{pq} la coloration de $x_1, \ldots, x_p$ avec q couleurs. De K_{pq} on déduit toutes les solutions qui préservent la coloration de $x_1, \ldots, x_p$ par la méthode suivante :

(1) Introduire un nouveau noeud x_{p+1} .

(2) Colorer x_{p+1} successivement avec les couleurs $1,2,\ldots q+1$

(3) Eliminer les cas qui ne constituent pas des colorations

(4) Continuer la même procédure avec toutes les nouvelles solutions partielles.

Toutes les solutions qui utilisent le nombre de couleurs minimum sont optimales.
Après cette définition formelle de l'arbre des solutions, RANDALL-BROWN donne un algorithme qui dépasse cet arbre, dans le sens où il réduit le nombre de solutions; celles-ci sont examinées tandis qu'on en mémorise une et une seule à un moment donné. Dans ce paragraphe, nous présentons deux algorithmes qui utilisent l'algorithme de RANDALL-BROWN, certaines idées contenues dans son article, et certaines idées nouvelles.

Avant cette présentation, remarquons que si nous colorons, successivement, les noeuds d'un graphe avec la plus petite couleur possible, dans un ordre donné, nous commençons évidemment la coloration d'une clique. Donc nous obtenons une borne inférieure du nombre chromatique. Il existe au moins deux méthodes qui donnent des cliques d'une dimension presque maximale, parce qu'elles utilisent la structure du graphe : la méthode de saturation et l'algorithme SL.

Soulignons, aussi, qu'une méthode séquentielle, qui colore chaque noeud avec la plus petite couleur possible, donne le chemin le plus à gauche dans l'arbre des solutions avec les sommets considérés dans l'ordre donné par l'ordre de coloration de la méthode en question.

Notons, enfin, que nous pouvons colorer les sommets d'une clique au début de la coloration sans augmenter le nombre de couleurs utilisées.

(III.3.1) ALGORITHME DE BROWN MODIFIE

Définition : On définit le rang d'un sommet comme la position de ce sommet dans l'ordre de coloration.

Algorithme (EM)

Soit G(X,E) le graphe à n sommetsque l'on doit colorer.
Soient w la dimension d'une clique initiale ;
q le nombre de couleurs utilisées par la méthode heuristique (SAT). Sans perte de généralité, supposons que $w < q$ et $x_1 \ldots x_w$ sont les sommets de la clique.

BEGIN initialiser back=false, k=w+1. Labeller tous les sommets de la clique

DO FOREVER
IF not back THEN

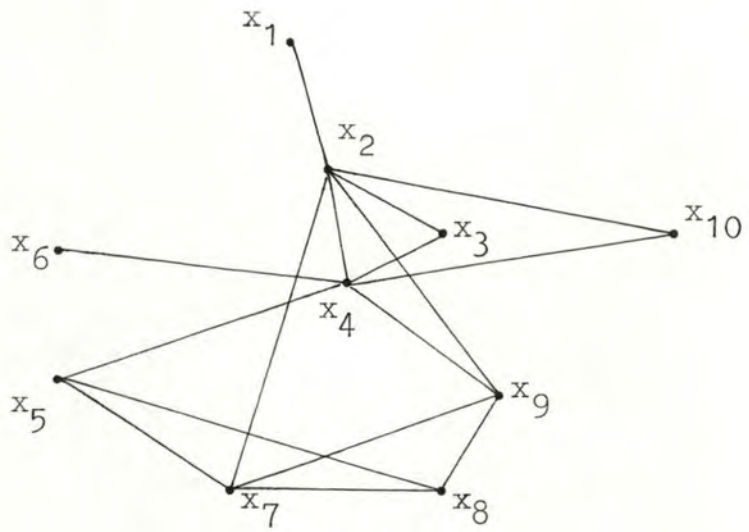
Déterminer u_K comme le nombre de couleurs utilisées par la solution partielle actuelle de niveau k-1, déterminer $U(x_k)$ comme l'ensemble de couleurs de $\{1,\ldots, \min(u_K+1, q-1)\}$ qui ne sont pas utilisées dans la solution partielle actuelle de niveau k-1 pour un adjacent de x_k

ELSE

Soit c la couleur actuelle de x_k et faire $U(x_k)=U(x_k) - \{c\}$, effacer le label de x_k si il y en a un.

```
    FI;
    IF U(x_k) ≠ ∅ THEN
        Déterminer i, la couleur minimale dans U(x_k); colorer
        x_k avec i ; (i est maintenant la couleur actuelle de
        x_k);k=k+1
        IF  k > n   THEN
                    On a trouvé une nouvelle solution; soit
                    s le nombre de couleurs utilisées par cette
                    solution; q = s ; EXIT IF  q=w;
                    déterminer k, le rang minimal parmi tous
                    les sommets q-colorés et effacer tous les
                    labels de x_k ... x_n
                    back = TRUE;
          ELSE
                    back = FALSE
          FI;
    ELSE
          back =    TRUE
    FI;
    IF back THEN
          Appeler la procédure label (x_k); déterminer k, le
          rang maximum parmi tous les sommets labellés
          EXIT IF k ≤ w
    FI;
OD;
STOP;
PROCEDURE Label (x_k);
          Labeller tous les sommets non labellés qui possèdent
          les propriétés suivantes :
          (i) de rang plus petit que le rang de x_k,
          (ii)adjacent à x_k
          (iii) de rang minimal parmi tous les sommets de
                leur couleur qui sont adjacents à x_k;
```

Exemple : Soit à colorer le graphe à 10 sommets suivant :



Les sommets x_2, x_4, x_9 constituent une clique. Supposons que l'ordre de coloration des noeuds soit le suivant : $x_2, x_4, x_9, x_7, x_5, x_8, x_3, x_{10}, x_1, x_6$.

La méthode de saturation donne la coloration initiale : 1,2,3, 2,1,4,3,3,2,1.

Cette coloration utilise 4 couleurs où q=4 et w=3. Si nous essayons d'améliorer cette coloration nous devons changer la couleur du sommet x_8. x_8 est adjacent à x_9, qui appartient à la clique; pour cette raison la couleur 3 est interdite au sommet 8. Pour les couleurs 1 et 2, x_8 n'est pas adjacent à un noeud de la clique. Les sommets de rang minimal sont x_7 (coloré 2) et x_5 (coloré 1); le rang maximal est celui de x_5. Affectons la couleur 3 à x_5 puisque celui-ci ne peut être coloré 2. Donc nous avons la coloration $x_5(3)$, $x_8(1)$; $x_3(3)$, $x_{10}(3)$, $x_1(2)$, $x_1(1)$. Le nombre de couleurs utilisées par la nouvelle solution est 3 qui est aussi la dimension de la clique; donc le nombre chromatique est 3.

Théorème 3 L'algorithme de BROWN modifié est une méthode exacte pour colorer les sommets d'un graphe.

Preuve - Premièrement, il est clair que dans toutes les situations où on affecte la valeur TRUE à la variable BACK, un backtrack est nécessaire. De ce fait, considérons la situation où il y a un backtrack de x_k à $x_{k'}$ avec $k > k'$. Nous devons démontrer qu'il n'existe pas de s-coloration S telle que $s < q$ et $S \supset [(x_1,c(x_1)), \ldots, (x_{k'},c(x_{k'}))]$, où cet ensemble représente la solution partielle actuelle de niveau k'.
Cette affirmation est vraie si $k = n$ et $x_{k'}$ est le sommet de rang minimum parmi les noeuds colorés q. Donc, les seuls backtracks qui sont intéressants sont ceux pour lesquels $x_{k'}$ est de rang maximal parmi tous les noeuds labellés. Soit m le nombre de ces backtracks et soit $L_m = \{(x_j,c(x_j)): x_j$ appartient à l'ensemble des sommets labellés quand le $m^{ième}$ backtrack est fait$\}$.

Nous allons maintenant montrer par induction sur m qu'il n'existe pas de solution S avec $s < q$ et $L_m \subset S$.

Si m=1, alors toutes les couleurs 1, ..., q-1 sont interdites pour x_k par un adjacent de x_k qui est déjà coloré avec une de ces couleurs. Donc, il est impossible de trouver une solution tant qu'un de ces noeuds n'a pas changé de couleur.

Si $m > 1$, nous avons deux situations à considérer :

(i) Identiquement au cas m=1, toutes les couleurs 1,...,q-1 sont interdites à x_k par un de ces adjacents .

(ii) Il existe une couleur $c < \min\{u_k+1, q-1\}$ qui n'est pas interdite à x_k par un adjacent coloré à l'aide de celle-ci, mais qui a été éliminée de $U(x_k)$ par un backtrack précédent $m' < m$; i.e. $(x_k,c) \in L_{m'}$. Si il existe une solution S avec $s < q$ et $L_m \subset S$, alors x_k doit avoir une couleur telle que c, i.e

$(x_k, c) \in S$. Ceci considéré avec $(x_k, c) \in L_m$, et $L_{m'} - \{(x_k, c)\} \subset L_m \subset S$ implique $L_{m'} \subset S$, qui est en contradiction avec notre hypothèse d'induction.

Etude de complexité de la méthode

Si on utilise la méthode du degré de saturation pour effectuer la première coloration sa consommation en temps calcul est $\delta(n^2)$. D'autre part, toutes les opérations comprises dans la boucle D O s'exécutent en $\delta(n)$ temps de calcul. Donc le problème est de savoir combien de fois cette boucle peut s'exécuter Pour déterminer ce nombre, soient q_o le nombre chromatique du graphe, w la dimension de la clique initiale, et q_1 le nombre de couleurs utilisées par la méthode heuristique, alors notre algorithme pourra tenter au plus :

$$\sum_{i=q_o}^{q-1} i^{n-w} \quad \text{colorations}$$

Le nombre cherché résulte donc proportionnel à q_o^{n-w}.

La complexité de l'algorithme de Brown modifié est de l'ordre de $\delta(n^2 + n\, q_o^{n})$.

(III.3.2.) ALGORITHME DE BROWN MODIFIE AVEC LOOK-AHEAD RULE

La technique du look-ahead consiste à déterminer si une couleur candidate $c_i \in U(x_k)$ causera, plus tard dans le processus de coloration une augmentation du nombre de couleurs nécessaires Cette information est caractérisée par deux nombres :

Premièrement par le "number of blockings" de la couleur c_i. Celui-ci est défini comme le nombre d'adjacents non colorés de x_k pour lesquels la couleur c_i est la seule disponible parmi le nombre de couleurs fixé. La signification de ce nombre repose sur le fait que, si nous colorons x_k avec une couleur c qui a un "blocking number" positif, alors nous sommes sûrs que nous ne pourrons finir notre coloration avec le nombre de couleurs donné. Nous savons que nous devrons utiliser une nouvelle couleur.

Le second nombre utilisé est le "number of preventions" qui est le nombre d'adjacents non colorés de x_k pour lesquels c_i est une couleur libre.
Ces deux notions sont utilisées pour déterminer quelle couleur $c_i \in U(x_k)$ sera assignée au noeud x_k.

Nous allons,maintenant, montrer comment on introduit la technique du look-ahead dans l'alorithme de Brown modifié.

Algorithme (EMLA)

Soit $w < q$ et $x_1, \ldots x_w$ définis comme au paragraphe précédent.

BEGIN

Initialiser back= false, block=false, k=w+1 ;
labeller tous les sommets de la clique

DO FOREVER

IF not back THEN

déterminer u_k et $U(x_k)$; pour chaque $c \in U(x_k)$ calculer le "blocking number" et le "number of preventions", ordonner les couleurs $c \in U(x_k)$, d'abord par rapport au "blocking number" et ensuite par le "number of preventions".

ELSE

Soit c la couleur actuelle de x_k; faire $U(x_k)=U(x_k)-\{c\}$;
effacer le label de x_k si il en existe un

FI;

IF $U(x_k) \neq \emptyset$ THEN

déterminer i la couleur de rang minimal dans $U(x_k)$;

```
                IF le "blocking number" de i est nul
                   THEN colorer le noeud x_k avec i; k=k+1
                        IF k > n   THEN
                                   On a trouvé une nouvelle
                                   solution; q=s;
                                   EXIT IF  q=w
                                   déterminer k, le rang mini-
                                   mal parmi tous les noeuds
                                   q-colorés et effacer les
                                   labels des sommets
                                   x_k,..., x_n;
                                   back  = true
                        ELSE
                                   back=false

                        FI;
                ELSE
                        back=true; block=true
                FI;
        ELSE
                back=true
        FI;
        IF back THEN
                IF block THEN
                        pour chaque couleur c ∈ U(x_k) de
                        "blocking number"positif : détermine
                        tous les sommets x_c qui seraient
                        bloqués par c et appeler la procédur
                        label (x_c) pour chacun d'eux;
                        block = false;
                FI;
                appeler la procédure label (x_k): déterminer k
                le rang maximal parmi tous les sommets labell
        OD;
        STOP;
        END;
```

Théorème 4 :

L'algorithme de Brown modifié avec le look-ahead rule est une méthode exacte pour colorer les noeuds d'un graphe.

Preuve : La démonstration est similaire à celle présentée au paragraphe précédent.

Etude de la complexité de l'algorithme

L'introduction du look-ahead rule induit une augmentation du temps calcul d'une itération de la boucle D O qui est maintenant de l'ordre de $\delta(n^3)$.

Toutefois, on espère, grâce à ceci, diminuer le nombre d'itérations par rapport à celui consenti par l'algorithme n'usant pas du Look-ahead rule; mais, sans pour cela, lui retirer son caractère exponentiel.

(III.4) - METHODE DE L'ENUMERATION IMPLICITE

Cette méthode a été mise au point dans le cadre de ce travail. Notre point de départ fut l'algorithme de l'énumération implicite approximative et nous l'avons transformé pour obtenir une méthode exacte. L'idée maîtresse de cette mutation était de permettre au processus de retrait de couleur à un sommet, d'envisager le changement de couleur, non plus pour un seul adjacent du sommet en question, mais de tous ces derniers. Dans cette optique plusieurs algorithmes ont été développés; nous avons retenu le plus performant de ceux-ci.

L'algorithme ENIMP est le suivant :

- Soit G (X,E) le graphe à colorer et notons par L la matrice de temps du problème.

  1) Trouver la dimension maximale des cliques du graphe. Ceci donnera une borne inférieure du nombre chromatique notée binchr.

  2) Trouver un ordre de coloration des noeuds, une coloration initiale avec q couleurs (une borne supérieure du nombre chromatique) en utilisant l'algorithme SAT.

  3) Si q = binchr alors stop.
  Sinon soit i^o le rang minimum parmi les noeuds q-colorés.

  4) Si $(L^o(i^o,j) = o \ \forall j < q) \vee (i^o=1)$ alors STOP

  Marquer tous les x_i t.q. $(i < i^o) \wedge (i,i^o) \in E$;
  Sélectionner k le rang maximum parmi les sommets marqués.

  Si k n'existe pas; alors STOP.
  Sinon . effacer la marque de x_k

  . soient j_k la couleur de x_k et $j'_k = \min \{ j' \text{ t q. } L^{k-1}(k,j') = 1) \wedge (j' \geq j_k + 1) \}$

  . Si $(j'_k \geq q) \vee (j'_k$ n'existe pas) alors $i^o = k$ aller en 4)

. Sinon colorer x_k avec j'_k ;
$L^k = L^{k-1}$; $L^k(i, j'_k) = o$ $\forall x_i$ adjacent à x_k ;
aller en 5)

5) effacer les marques de tous les sommets
(colorer chacun des sommets x_i restants ($i > k$) avec la plus petite couleur possible $< q$)

Pour i de k+1 à n faire

- soit $j = \min \{ j' \text{ t.q } L^{i-1}(i,j') = 1 \}$
- Si $j \geq q$ alors $i^o = i$ aller en 4)
- Sinon $f(x_i) = j$; $L^i = L^{i-1}$; $L^i(i', j) = o$ $\forall x_{i'}$ adjacent de x_i

Fin faire

On a obtenu une nouvelle solution utilisant $s < q$ couleurs;
$q = s$
aller en 2)

REMARQUE : Dans une dernière tentative pour améliorer la performance de notre algorithme nous décomposions le graphe en composantes connexes et, pour chacune de celles-ci nous calculions la dimension maximale de leurs cliques, le maximum de ces nombres donnant une borne inférieure du nombre chromatique. Les étapes 2) à 5) étaient alors appliquées aux différentes composantes connexes prises séparément. En procédant de cette manière nous espérions diminuer le temps calcul consommé en traitant plusieurs sous-problèmes de tailles réduites plutôt que le graphe entier.
Malheureusement, les tests ne se sont pas révélés concluants.

ÉTUDE DE COMPLEXITE

. l'étape 1) n'est exécutée qu'une fois mais requiert comme nous l'avons vu précédemment $\mathcal{O}(n^2.(2^{\mu.n + \lfloor \frac{2}{\mu} \rfloor}))$ temps calcul (où μ est la densité du graphe)

. l'étape 2) est exécutée également une fois et ne coûte que $\delta(n^2)$ temps calcul (cfr.(II.2))

. l'étape 3) exige , quant à elle, $\delta(n)$ temps calcul

. le temps calcul consommé entre l'activation et la terminaison d'une étape 4) n'excède pas $\delta(n^2)$. Finalement, les opérations de recoloration dictées par l'itération 5) coûtent également $\delta(n^2)$ temps calcul.

La complexité de la procédure sera, maintenant, déterminée par le nombre d'exécutions des étapes 4) et 5). Ce nombre est évidemment égal à celui des recolorations tentées par l'algorithme et est de l'ordre de $\delta(q^n)$ (où q est le nombre chromatique du graphe). Donc l'algoritme ENIMP requiert $\delta(n^2 [\, 2^{\mu} n + |\frac{E}{\mu}| + q^n \,])$ temps de calcul.

(III.5) RESULTATS DE TESTS

Dans le même contexte (ordinateur et langage) que leurs homologues non-exactes, les méthodes présentées dans ce chapitre ont donné lieu à une implémentation. Celle-ci avait pour but de soumettre les méthodes exactes à des jeux de test identiques à ceux dont il est fait mention en (II.4).

Dans ce cas, la seule performance observée fut le temps de calcul.
A cet effet, les figures (III.3) et (III.4) offrent, respectivement, un tableau regroupant les différents résultats obtenus pour les 16 classes de graphes proposés à la coloration et un graphique attestant de l'évolution du temps CPU consommé par chaque méthode en fonction de la taille du graphe à colorer.

CONCLUSION DES TESTS

Des tests numériques effectués, on peut tirer les conclusions suivantes :

1) Nous ne fournissons aucun résultat pour la méthode de N. CHRISTOFIDES . En effet, lors de tests préliminaires, celle-ci a consommé 6h.44 min. de temps CPU pour colorer un graphe de 20 sommets dont la densité valait 0.5 ! Ceci nous a évidemment dissuadé de la soumettre aux tests prévus.

2) Malgré les améliorations que C. WANG a apporté à cette dernière méthode (cfr. (III.2)), son algorithme n'est guère applicable que pour de petits graphes. Ceci est une conséquence, non seulement du temps de calcul exigé, mais aussi de la place mémoire requise pour stocker les ensembles indépendants maximaux du graphe. Notons aussi que pour des graphes de taille fixe les réquisitions en ressources (CPU et mémoire) de cet algorithme croissent au fur et à mesure que la densité décroît. Ceci peut être expliqué en remarquant que, si on fait tendre la densité d'un graphe vers zéro, le nombre d'ensembles indépendants maximaux de celui-ci augmente et il en est de même pour le nombre de noeuds de l'arbre des r-sous-graphes construit par la méthode de WANG.

3) Soulignons les résultats intéressants obtenus par les algorithmes de Brown modifiés, vu le caractère exponentiel du problème du nombre chromatique d'un graphe. Notons aussi, que la procédure "Look-ahead" est certainement utile pour les densités faibles. Pour les autres graphes, les résultats de cet algorithme ne sont pas si bons : une efficacité identique ou inférieure à la méthode qui n'utilise pas le "look-ahead" .

4) Quant à notre méthode, l'énumération implicite, si ses performances sont largement supérieures à celles de l'algorithme proposé par C. WANG, elle ne bénéficie cependant pas d'un comportement aussi appréciable que les méthodes de Brown modifiées. Ceci peut être expliqué, en partie, par l'utilisation, de la part de notre méthode, de l'algorithme de recherche des cliques maximales d'un graphe qui, même si elle fournit une borne inférieure du nombre chromatique plus précise, consomme beaucoup de temps calcul. Signalons toutefois que notre algorithme obtient un résultat supérieur à celui des méthodes de Brown modifiées pour les graphes de la classe XV.

5) Finalement, ce sont les méthodes de Brown modifiées qui ont procuré les meilleures performances.

N° de classe	Temps CPU consommé (en sec.)			
	Wang	em	em la	enimp
I	3441.11	0.04	0.03	0.11
II	9.32	0.06	0.05	0.19
III	0.80	0.05	0.04	0.73
IV	0.45	0.05	0.04	0.62
V	-	0.64	0.38	59.80
VI	-	2.78	2.20	49.44
VII	-	0.42	0.32	793.10
VIII	92.61	0.29	0.22	21.97
IX	-	4.73	2.28	2213.19
X	-	-	-	-
XI	-	4877.42	1800.18	-
XII		5.48	5.51	2603.27
XIII	-	2.68	2.43	-
XIV	-	-	5781.98	-
XV	-	-	-	116.75
XVI	-	1.44	-	78.17

Fig. (III.3). ' - ' = plus de 2 h CPU

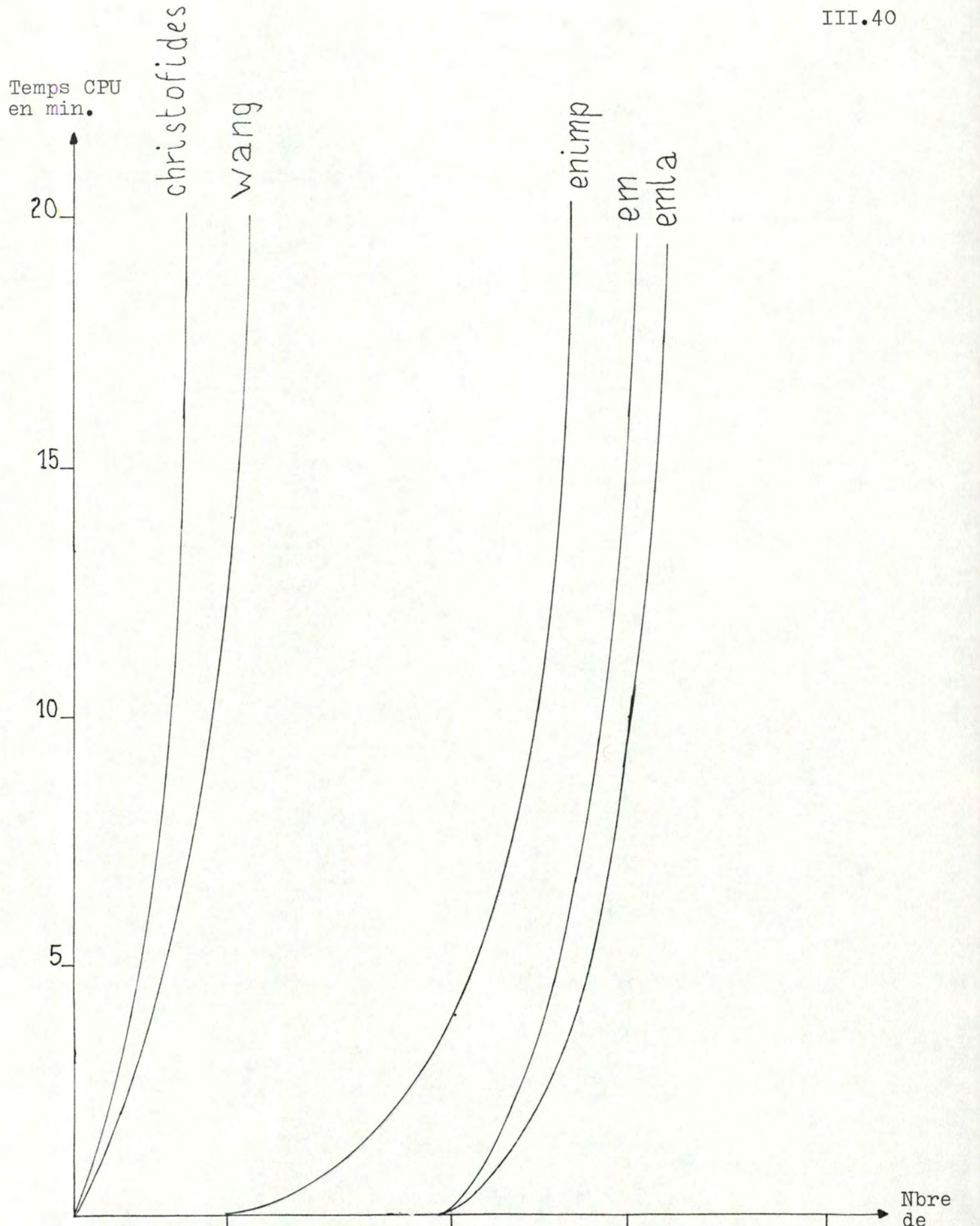

Temps CPU
en min.
20
15
10
5
christofides
wang
enimp
em
emla
20
50
75
100
Nbre
de
sommet

Fig. (III.4)

CHAPITRE IV. - POINTS DELICATS DE L'IMPLEMENTATION

CHAPITRE IV : POINTS DELICATS DE L'IMPLEMENTATION

Nous levons, ici, le voile sur quelques détails de l'implémentation des méthodes présentées dans les chapitres II et III.

(IV.1) - IMPLANTATION D'UN GRAPHE EN MEMOIRE

Comme nous l'avons vu lors des préliminaires, un graphe peut être représenté de manière unique par sa matrice d'adjacence. Hélas, la densité des graphes rencontrés dans la pratique n'excèdant en général pas 0.25, cette dernière structure risque d'être souvent creuse. Pour économiser de la place mémoire, nous utiliserons, pour implémenter un graphe, les vecteurs CI et CL de la manière suivante :

CI sert d'index à CL, la liste des sommets adjacents.

Par exemple, les sommets adjacents du $i^{ième}$ noeud sont stockés séquentiellement dans CL(CI(i-1)+1),...,CL(CI(i))

(IV.2) - INTERDICTION D'UNE COULEUR AUX ADJACENTS D'UN SOMMET DONNE

Le problème est rencontré, essentiellement, dans les méthodes séquentielles par sommets et les méthodes de saturation.
On pourrait, à cet effet, utiliser une matrice $L=(L_{ij})$ telle que

$$L_{ij} = \begin{cases} 0 & \text{Si le sommet i peut être coloré avec j} \\ 1 & \text{sinon} \end{cases}$$

La $i^{ième}$ ligne de la matrice L est donc une suite constituée de 0 et de 1. De ce fait, nous pouvons considérer cette $i^{ième}$ ligne comme la représentation binaire d'un nombre entier. Il nous aurait, alors, été loisible de prendre une variable entière pour stocker le nombre entier indiquant les couleurs interdites

au sommet i; mais, en FORTRAN 77, les entiers sont codés sur 32 bits (dont 1 bit de signe) et donc, une telle formule nous permettrait d'interdire seulement 31 couleurs à un sommet. En vue d'envisager un plus grand nombre de couleurs nous allons utiliser 3 variables de la manière suivante :

La première variable indiquera les couleurs disponibles de 1 à 31
La deuxième " " " " de 32 à 62
La troisième " " " " de 63 à 93

Le problème général de tous les sommets sera résolu en considérant un vecteur L tel que les composantes L(3i-2),L(3i-1),L(3i) indiquent les couleurs disponibles pour le noeud i.

(IV.3) IMPLEMENTATION DE LA METHODE "ADJACENTS DES ADJACENTS"

Nous utiliserons des vecteurs E et F pour repérer à tout instant du programme l'état du graphe G_i (sommets colorés, nombre d'adjacents, ..) et des ensembles d'adjacents A_j :

$$\begin{cases} E(k) = \text{nombre d'adjacents de k dans le graphe } G_i \text{ si k est non coloré et } k \notin \bigcup_l A_l \\ E(k) = -1 \text{ si k est coloré et } k \notin \bigcup_l A_l \\ E(k) = -(j+1) \text{ si } k \in A_{j-1} \end{cases}$$

$$\begin{cases} F(k) = -1 \quad \text{Si k est coloré} \\ F(k) = \text{nombre d'adjacents de k dans le graphe } G_i \end{cases}$$

La coloration d'un sommet peut se faire à 3 endroits différents du programme : (cfr § II.4.1)

a) en 3): colorer le sommet k de degré maximal dans la composante connexe traitée de G_i ; ceci entraîne

(i) la mise-à-jour du sous-graphe suivant (qui ne tient pas compte des sommets colorés du graphe actuel)

$$(IV.1)\begin{cases} F(k) = -1 \\ F(k') = F(k')-1 \quad \forall\ k' \text{ adjacent de } k \end{cases}$$

(ii) marquer que le sommet $k \in A_{j-1}$ et que ses adjacents dans G_i appartiennent à A_j .

$$(IV.2)\begin{cases} E(k) = -(j+1) \\ E(k') = -(j+2) \quad \forall \text{ adjacent } k' \text{ de } k \end{cases}$$

b) en 7) colorer tous les sommets k possibles de A_{j+1} par degrés décroissants (à l'aide de F(k))

$$\begin{cases} F(k) = -1 \\ F(k') = F(k')-1 \quad \forall\ k' \text{ adjacent à } k \end{cases}$$

c) en 10) coloration des sommets isolés k de G_i ; il suffit de les marquer vu le fait qu'ils n'ont pas d'adjacents

$$F(k) = -1$$

Pour garnir l'ensemble A_{j+1} , on procède comme suit :

parcourir l'ensemble E : l , $1 \leqslant l \leqslant n$

$$(IV.3)\begin{cases} \text{si } (E(l) \neq -1) \text{ et } (E(l) > -(j+1)), \\ \text{alors si } \exists \text{ 1 adjacent } l' \text{ de } l \text{ tel que } E(l')=-(j+2) \\ \qquad \text{alors } E(l) = -(j+3) \text{ (i.e. } l \text{ appartient à } A_{j+1}) \end{cases}$$

Nous définissons les procédures DELETE et ADJACA pour effectuer les traitements (IV.1) et (IV.2).

- DELETE (D,l) : Pour un vecteur D et un noeud l donnés, cette procédure exécute les opérations suivantes :

$$D(l) = -1$$

$$D(l') = D(l')-1 \quad \forall\, l' \text{ adjacent de } l$$

- ADJACA (D,l,j) : Pour un vecteur D, un noeud l, un entier ≥ 0 j, donnés, cette procédure exécute les opérations suivantes :

$$D(l) = -(j+1)$$

$$D(l') = -(j+2) \quad \forall\, l' \text{ adjacent de } l \text{ t.q. } D(l') > 0$$

Ces 2 procédures utilisent $\Theta(d)$ temps de calcul où d est le degré moyen de chaque sommet du graphe.

Un sommet coloré avec la couleur i a été traité

- 1 fois par DELETE
- au plus (i-1) fois par ADJACA
- au plus i fois par (IV.3) (remarquons que le temps calcul de (IV.3) est $\Theta(d)$)

donc pour l'ensemble des sommets du graphe (k étant le nombre total de couleurs et n_i le nombre de sommets colorés i)

$$\sum_{i=1}^{k} n_i \,(1 * \Theta(d)+(i-1)*\Theta(d)+i * \Theta(d)) = \sum_{i=1}^{k} n_i * \Theta(d)$$

$$*i \leq \sum_{i=1}^{k} k*n_i * \Theta(d) = k \sum_{i=1}^{k} n_i * \Theta(d) = k*n* \Theta(d)$$

nous donne donc $\Theta(k * n * d)$ de temps de calcul. Les opérations restantes (recherche du sommet de degré maximal,...)

sont faites en $\delta(n^2)$ temps de calcul. Donc l'algorithme exige $\delta(n^3)$ temps de calcul.

Si par contre $k * d = \delta(n)$ où $k.u = \delta(n^2)$, u étant le nombre d'arêtes du graphe, alors le temps de calcul sera $\delta(n^2)$. Puisque cette propriété est vérifiée par les graphes de grande taille rencontrés dans la pratique, ces derniers peuvent être colorés par la méthode "adjacents des adjacents" en $\delta(n^2)$ temps de calcul.

(IV.4.) IMPLEMENTATION DE L'ALGORITHME RLF (/2/)

Comme il a été démontré dans l'exemple du paragraphe (II.4.2) l'algorithme s'applique facilement aux petits graphes. Pour faciliter la tâche pour des graphes de grande taille, d_{X_1} et d_{X_2} doivent être stockés pour chaque sommet et être modifiés lors de changements dans X_1 ou X_2. Ceci sera fait par les vecteurs E et F :

$$E(x) = \begin{cases} < 0 & \text{si } x \text{ est coloré} \\ < 0 & \text{si } x \in X_2 \\ = d_{X_1}(x) & \text{si } x \in X_1 \end{cases}$$

et

$$F(x) = \begin{cases} < 0 & \text{si } x \text{ est coloré} \\ = d_{X_1 \cup X_2}(x) & \text{si } x \in X_2 \\ = d_{X_1 \cup X_2}(x) & \text{si } x \in X_1 \end{cases}$$

Initialement X_1 contient tout sommet de G et donc $E(x) = F(x) = d(x)$ $\forall x \in X$. Si le sommet x' est sélectionné pour la coloration, alors on modifie E et F tels que

$$\begin{cases} E(x') = -1 \\ E(x) = -1 \text{ pour tout adjacent } x \text{ de } x' \\ E(x) = E(x) - (\text{le nombre de sommets de } X_1 \text{ adjacents à } x \text{ et } x') \text{ pour } x \in X_1 \text{ non-adjacent à } x' \\ F(x') = -1 \\ F(x) = F(x)-1 \text{ pour tout adjacent } x \text{ de } x' \end{cases}$$

Le sommet suivant à colorer de X_1 est celui ayant une valeur maximale de F(x)-E(x) et, si une égalité existe, celui à valeur de E(x) minimale.
Si X_1 est vide, (i.e. $E(x) < o, \forall\ x \in X$), les valeurs E sont modifiées t.q. $E(x) = F(x)\ \forall\ x \in X$. Ceci correspond à une réinitialisation des valeurs de E pour le sous-graphe de G engendré par les sommets non colorés.

L'implémentation des opérations sur E et F est facile en utilisant la sous-routine décrite au paragraphe précédent DELETE.
Donc, si x' est sélectionné pour une coloration, E et F seront modifiés par un appel à DELETE appliqué sur (F,x') et (E,x') ainsi que sur (E,x) $\forall x \in X_1$ adjacent à x'. Il est facile de vérifier qu'une telle procédure maintient la valeur désirée de E et F.

Tout sommet x du graphe est traité par la routine DELETE f(x) fois (où f(x) est la couleur de x). En effet, en observant que pour toute nouvelle couleur i introduite, $i \leqslant f(x)$, x est soit coloré i, soit adjacent à un sommet coloré i.
Dans chaque cas x sera effacé exactement une fois.
Dès qu'il sera coloré, il ne sera par conséquent plus effacé.

Donc $\sum_{i=1}^{k} i.n_i \leqslant \sum_{i=1}^{k} k.n_i = k.\sum_{i=1}^{k} n_i = k.n$ effacements

(appels à DELETE).

seront faits sur G, où k est le nombre de couleurs utilisées, n_i est le nombre de sommets colorés i, n le nombre de sommets de G.
Comme chaque effacement demande $\delta(d)$ temps de calcul, où d=degré moyen de tout sommet, tous les effacements peuvent être faits en $\delta(kdn)$ temps de calcul.
Il est évident que les opérations restantes (calcul des adjacents,...) exigent $\delta(n^2)$ temps de calcul, donc l'algorithme réclame au total $\delta(n^3)$ de temps. Pourtant, pour des graphes tels que $kd \simeq \delta(n)$, i.e. $k * u = \delta(n^2)$, où u est le nombre d'arêtes du graphe, l'algorithme RLF exige $\delta(n^2)$ de temps. Cette propriété est vérifiée pour beaucoup de graphes de grande taille et ceci souligne dès lors l'efficacité de cette méthode de coloration.

(IV.5) IMPLEMENTATION DE LA RECHERCHE DES CLIQUES MAXIMALES

Dans le but d'implémenter cette méthode en FORTRAN, il nous faut éliminer la recursivité contenue par celle-ci. Pour ce faire, on a recours à une pile pour stocker les différents graphes $\tilde{G}$ et les ensembles de noeuds N associés. L'algorithme non récursif s'écrit alors :

1) $\tilde{G} = G$, $\tilde{X} = X$, $N = \emptyset$, pile vide, $S = \emptyset$

2) Si $\tilde{X} \subseteq \Gamma(x_i)$ pour $x_i \in N$ aller en 5)
sinon aller en 3)

3) Si $\tilde{G}$ est complet alors $S = S \cup \tilde{X}$ aller en 5)
sinon aller en 4)

4) choisir x_k dans $\tilde{G}$ qui n'est pas adjacent à tous les noeuds de $\tilde{X}$.
Mettre $(\tilde{G},N)$ dans la pile
Construire le sous-graphe $\tilde{G}_k$ de $\tilde{G}$ qui est engendré par x_k et ses adjacents
$\tilde{G} = \tilde{G}_k$
aller en 2)

5) Si la pile est vide - STOP
sinon sortir $(\widetilde{G},N)$ de la pile
Construire $\widetilde{G}_{\overline{k}}$ et $N_{\overline{k}}$ comme indique en (III. 1.4)
$\widetilde{G} = \widetilde{G}_{\overline{k}}$ et $N = N_{\overline{k}}$
aller en 2)

La structure de pile est organisée de la manière suivante :

En fait, nous disposons de deux piles : une pile de tableaux et une pile de variables. La pile de tableaux sert à stocker, la représentation de l'ensemble de noeuds N. Celui-ci est implémenté à l'aide d'une variable entière qui dénombre les sommets appartenant à N et d'un vecteur qui contient ces différents noeuds. Dans la pile de variables on stocke une paire de valeurs, l'une étant le niveau de profondeur dans l'arbre, l'autre le sommet x_k considéré.

En outre, pour caractériser, à tout instant, le graphe $\widetilde{G}$, on utilise les vecteurs :

$$X(\omega) = \begin{cases} o & \text{ssi } \omega \in \widetilde{G} \\ i > o & \text{niveau où } \omega \text{ a été éliminé de } \widetilde{G} \end{cases}$$

$$DEG\ (\omega) = \begin{cases} \text{nombre d'adjacents dans } \widetilde{G} & \text{si } \omega \in \widetilde{G} \\ < o & \text{sinon} \end{cases}$$

On dispose également d'une variable indiquant le niveau de l'arbre auquel on situe.
La construction de $\widetilde{G}_k$ dans l'étape 4) est réalisée au moyen de l'application de la procédure DELETE, (cfr § (IV.3)), aux paires $(DEG, x_{k'})$ pour tout $x_{k'}$ non adjacent de x_k. De plus, pour ces mêmes $x_{k'}$, on affecte à la composante correspondante de X le niveau d'arborescence courant.
Dans l'étape 5) pour retrouver $\widetilde{G}$ à l'aide du sommet x et du niveau i, retirés de la pile, on fait subir à X et DEG les transformations suivantes :

pour tout ω t.q. $i \leq X(\omega) \leq$ niveau courant :

- DEG $(\omega) = o$
- $X(\omega) = o$
- pour tout adjacent ω' de ω t q $X(\omega')=o$

  DEG $(\omega') =$ DEG $(\omega')+1$

  DEG $(\omega) =$ DEG $(\omega)+1$

La construction de $\tilde{G}_k$ est alors simplement réalisée grâce à l'application de la routine DELETE à la paire (DEG,x), et à l'affectation $X(x) = i$.
Pour terminer, remarquons que, si le choix d'un x_k dans $\tilde{G}$ est effectué de telle manière que le sommet sélectionné soit de degré minimum dans $\tilde{G}$ alors l'ensemble des noeuds x_1 vérifiant la condition(III.9) du théorème 1, § (III.1.4), est considérablement réduit.

(IV.6) IMPLEMENTATION DES r-SOUS-GRAPHES MAXIMAUX DANS LA METHODE DE CHRISTOFIDES

On se donne 5 vecteurs R, I, Father, SI,S.
SI est un vecteur d'index de S qui contient les noeuds des différents $S_1(G^j)$ (où $G^j = \langle X-S_r^j(G) \rangle$, $j = 1,\ldots,q_r$, $r = 1,2, \ldots$). Donc les noeuds du $i^{\text{ième}}$ ensemble $S_1(G^j)$ sont stockés séquentiellement dans S(SI(i-1)+1), S(SI(i-1)+2), ..., S(SI(i)).
Chaque élément de I est un "pointeur" vers un des $S_1(G^j)$:
La composante I(k) désigne l'ensemble $S_1(G^j)$ composé par les noeuds S(SI(I(k)-1)+1), ..., S(SI(I(k)).
La formule de récurrence (III.1) nous indique qu'un r-sous-graphe maximal n'est rien d'autre qu'un regroupement d'ensemble $S_1(G^j)$. Les $S_1(G^j)$ d'un même r-sous-graphe maximal sont chaînés grâce au vecteur Father.

Finalement, R est un vecteur d'index de I. Ainsi, les premiers maillons des chaînes de $S_1(G^j)$ des r-sous-graphes maximaux sont représentés par I(R(r-1)+1), ..., I(R(r)).

On manipule cette structure à l'aide de deux procédures :

. EXTR accepte comme données les vecteurs S, SI, I, Father, et l'entier i et fournit un vecteur d'indicateurs u défini comme suit :

$$u(k) := \begin{cases} 1 & \text{si le noeud } x_k \text{ appartient au r-sous-graphe maximal dont le premier maillon de sa chaîne de } S_1(G^j) \text{ est désigné par } I(i). \\ o & \text{sinon} \end{cases}$$

. INTER accepte comme données les vecteurs S, SI, U et les variables nemid (nombre de $S_1(G^j)$ déjà présents dans S) i_1 et i_2 (t.q. $U(i_1)$, $U(i_1+1)$, ..., $U(i_2)$ contiennent les noeuds d'un $S_1(G^j)$) et affecte à la variable index, la valeur :

$$\begin{cases} k \leqslant \text{nemid} & \text{si } S(SI(k-1)+1) = U(i_1); \ldots S(SI(k)) = U(i_2). \\ \text{nemid} + 1 & \text{si il n'existe pas de k vérifiant la propriété exhibée ci-dessus. De plus le } S_1(G^j \\ & \text{contenu dans U est ajouté à S et SI; et} \\ & \text{nemid est incrémenté de 1.} \end{cases}$$

CHAPITRE V. - CONCLUSIONS

CHAPITRE V.- CONCLUSIONS

Sachant que le problème de coloration des graphes est NP-complet, nous nous sommes surtout intéressés au comportement réel des algorithmes qui résolvent exactement ce problème. Dans cette optique nous proposons un tel algorithme, dont les performances sont acceptables compte tenu de ce qui vient d'être mentionné.

Cet algorithme ainsi que 4 autres méthodes exactes et 20 heuristiques ont été implémentées en FORTRAN 77 sur VAX/VMS dans le but d'être testées sur quelque 80 graphes générés aléatoirement. C'est vraisemblablement la première fois que des tests mettent en question un si grand nombre de méthodes. De plus, lors de ces derniers, la procédure décrite en annexe 1, pour générer des graphes aléatoires dont le nombre chromatique est connu, est utilisée pour déterminer non seulement la précision, mais encore les capacités relatives des algorithmes étudiés.
Auparavant, beaucoup de tests comparatifs publiés étaient opérés sur des graphes dont le nombre chromatique était inconnu; ce qui rendait impossible toute évaluation de la précision d'un algorithme individuel et suspecte toute affirmation concernant les capacités relatives de 2 algorithmes.

Le graphique de la fig. (V.1) résume les résultats de ces tests présentés en fin des chapitres II et III. Ce graphique représente l'évolution du temps calcul consommé par quelques méthodes en fonction du nombre de noeuds du graphe à colorer. De celui-ci, nous pouvons déduire que les méthodes exactes ne sont raisonnablement applicables qu'à de petits graphes. Malheureusement, la taille des graphes rencontrés dans la pratique excède les 100 sommets, taille à laquelle nous nous sommes limités pour nos tests. A titre d'exemple, citons le graphe représentant le problème d'un calcul d'horaire

aux Facultés de Namur (Facultés de Sciences Economiques et Sociales) qui comporte 400 sommets (cfr. / 10/)!
Ceci exclut dès lors les méthodes exactes et impose l'utilisation d'algorithmes heuristiques. Parmi ces derniers, ceux qui fournissent l'approximation la plus proche du nombre chromatique sont les méthodes de saturation, l'algorithme RLF et les méthodes séquentielles munies de la permutation bichromatique. Cependant, la permutation bichromatique consomme plus de temps que les 2 autres méthodes citées; ce qui nous fait préférer ces 2 dernières. Enfin, puisque pour la plupart des applications pratiques grandes échelles, la densité d'arêtes du graphe à colorer est généralement faible et vu l'adéquation, déjà signalée, entre ces problèmes et RLF, du point de vue du temps calcul, nous conseillons cet algorithme d'autant qu'il produit des colorations un peu meilleures que les méthodes de saturation.

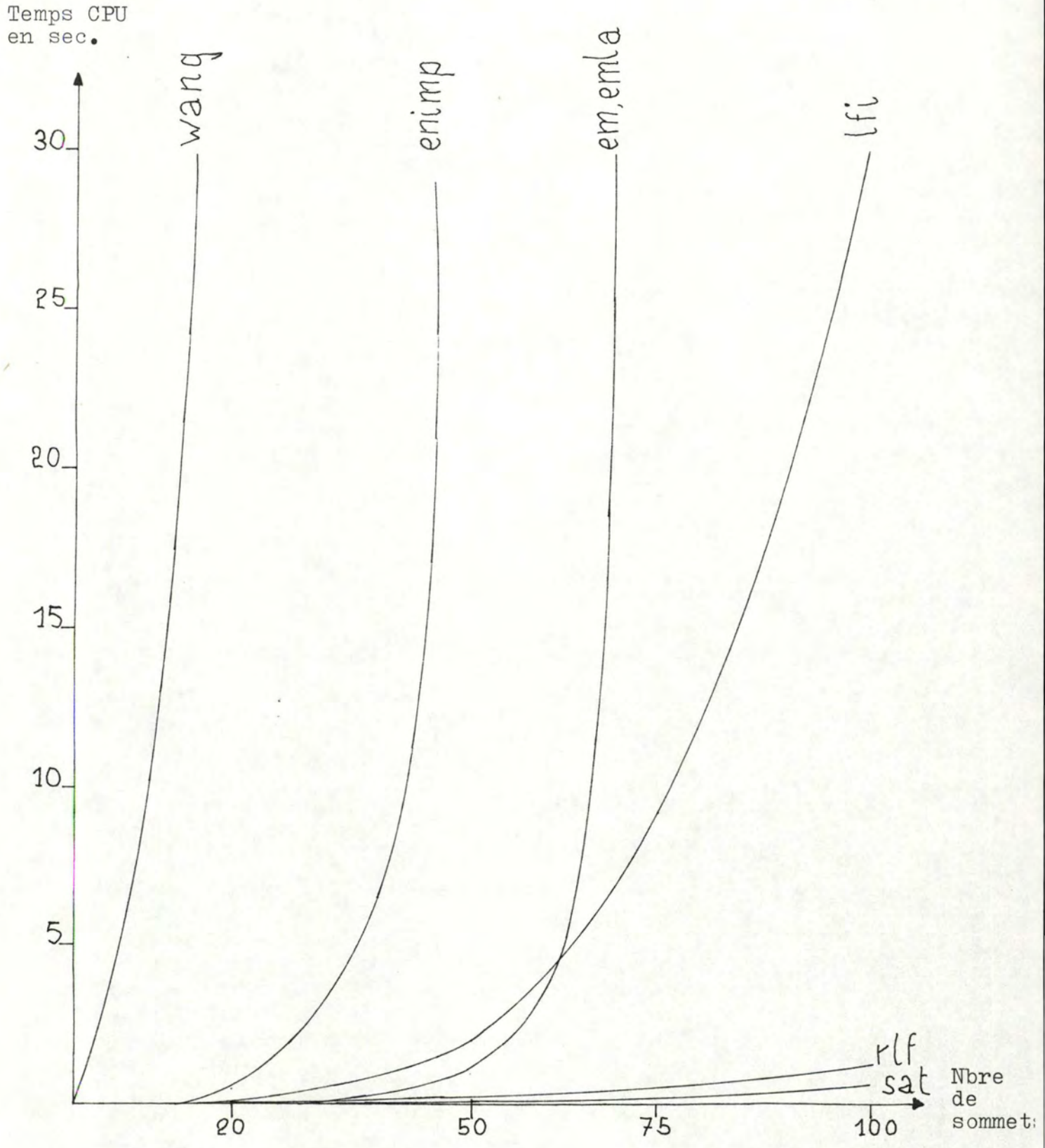

Temps CPU
en sec.
wang
enimp
em,emla
lfi
30
25
20
15
10
5
rlf
sat
20
50
75
100
Nbre
de
sommets

Fig. (V. 1)

ANNEXES : I. Génération de graphes à nombre chromatique connu

II. Tableaux résumant les performances des méthodes non-exactes

Génération de graphes à nombre chromatique connu (/2/)

Nous allons décrire un procédé de construction de graphes dont on connaît le nombre chromatique exact. Ceci permettra de mieux évaluer les algorithmes de coloration appliqués à ces graphes.

Supposons que l'on désire construire un graphe G(X,U) possédant n sommets, u arêtes et dont le nombre chromatique vaut k. En plus supposons que $k|n$ c'est-à-dire k divise n. Cette restriction n'est pas significative car la plupart des tests qui utilisent des générateurs de graphes offrent une certaine flexibilité quant aux choix de k et de n. Pour qu'un tel graphe existe, il faut que u satisfasse la condition suivante :

$$\frac{k(k-1)}{2} \ll u \ll \frac{n^2(k-1)}{2\,k} \qquad (1)$$

En effet, il est évident que $u \geqslant \frac{k(k-1)}{2}$ (le cas le plus favorable est une k-clique); de l'autre côté, la situation avec le plus d'arêtes est d'avoir $\frac{n}{k}$ k-cliques reliées entre elles sans former de (k+1)-cliques ce qui implique d'avoir :

$$\frac{k(k-1)}{2} \cdot \frac{n}{k} + \frac{(k-1)}{k} \cdot \frac{n(n-k)}{2} = \frac{n^2(k-1)}{2\,k}$$

arêtes; d'où (1).

Nous allons dans la suite décrire la procédure pour réaliser ce graphe G à nombre chromatique k.

Le premier pas de cette procédure est de choisir les entiers positifs a, n, c et m tels que :

1) $m \gg n$.

2) $(n,m) = k$ c-à-d le PGCD de n et m est k

3) $(c,m)=1$ c-à-d c et m sont premiers entre eux

4) $p \mid m \Rightarrow p \mid (a-1)$ pour tout nombre premier p et

5) $4 \mid m \Rightarrow 4 \mid (a-1)$

Ensuite, générons une suite de nombres aléatoires $\{X_i\} \subseteq [o,m-1]$ par la méthode congruentielle aléatoire (cfr /3/). Pour cela, fixons X_o et pour chaque $i > o$ calculons $X_i = \text{Mod}\ (a X_{i-1} + c, m)$; où $\text{mod}\ (X,Y) = X - \left[\frac{X}{Y}\right] \cdot Y$.
Les séquences ainsi générées jouissent de deux propriétés très importantes.

a) $\forall\ i,j$ t.q. $o \leqslant j \leqslant m-1$, $i \geqslant o\ \exists r$ t q $i \leqslant r \leqslant i + m-1$ $\wedge\ X_r = j$ c-à-d on retrouve chaque nombre une fois par période

b) $\forall\ i \geqslant o\ \ X_i = X_{i+m}$ c-à-d la période est m .

Les choix de 3) et 4) sont nécessaires pour que le générateur de nombres aléatoires obtienne une période maximum m.

Considérons, alors la suite $\{Y_i\} \subseteq [o,n-1]$ telle que

$$Y_i = \text{mod}\ (X_i,n) \qquad i = 1,\ldots,m$$

Remarquons que cette suite Y_i n'est pas une suite uniforme sur $[o,n-1]$ puisque $(n,m) = k \neq n$.

Soit $X = \{o,1,\ldots,n-1\}$, il est dès lors possible d'associer deux valeurs consécutives de $\{Y_i\}$ à des arêtes de U. De façon similaire, il est possible d'associer h valeurs consécutives de $\{Y_i\}$ à des cliques de G. Par exemple, la sous-suite $\{Y_1,Y_2,Y_3\}$ correspond au sous-ensemble de U: $\{(x_{Y_1},x_{Y_2})$, (x_{Y_1},x_{Y_3}), $(x_{Y_2},x_{Y_3})\}$. En considérant certaines sous-suites de $\{Y_i\}$ et en y ajoutant les arêtes correspondantes de U, il sera possible de construire le graphe souhaité G(X,U).

Plus précisément, soit le vecteur, de dimension k - 1,
$\vec{b} = \{b_k, b_{k-1}, \ldots, b_2\}$ tel que $b_k \geq 1$ et $b_i \geq 0$, $2 \leq i \leq k-1$.
Chaque b_i correspond au nombre de i-cliques implantées dans G.
En particulier, étant donné la séquence $\{Y_i\}$ et le vecteur $\vec{b}$ procéder de la façon suivante :

- choisir les k premières valeurs de $\{Y_i\}$ en commençant par Y_1 et ajouter les arêtes correspondantes à U (au début, U = $\emptyset$).
- Si $b_k > 1$, sélectionner les k valeurs suivantes de $\{Y_i\}$ et ajouter les arêtes correspondantes à U . Répéter ce processus jusqu'à avoir ajouté b_k k-cliques à G.
- de façon similaire, ajouter b_{k-1} (k-1)-cliques à G. Continuer jusqu'à l'ajout de b_2 2 -cliques c-à-d arêtes à U.

Remarquons que certaines arêtes peuvent être "ajoutées" plusieurs fois et il sera donc impossible de prévoir un vecteur $\vec{b}$ tel que le nombre total d'arêtes vaut exactement u.
Il est par contre possible de contrôler le nombre d'arêtes additionnées à un sommet donné et d'éliminer l'ajout de i-cliques résultant de l'ajout de trop d'arêtes à U. Comme les arêtes sont ajoutées une par une, il est facile de prouver qu'un graphe possédant exactement u arêtes peut être construit ainsi pour u tel que

$$\frac{k\,(k-1)}{2} \leq u \leq \frac{n^2(k-1)}{2\,k}$$

Il nous reste maintenant à prouver que $\chi(G) = k$ pour tout graphe G construit de cette manière. Puisque $b_k > 1$, G contient une k-clique, et donc $\chi(G) \geq k$. Avant de prouver que $\chi(G) \leq k$, examinons la structure de la séquence $\{Y_i'\}$; où $Y_i' = \mathrm{mod}\,(Y_i\ ,\ k)$

$$
\begin{aligned}
Y'_{i+1} &= \text{mod } (Y_{i+1}\ ,\ k) \\
&= \text{mod } (\text{mod}(X_{i+1}\ ,\ n)\ ,\ k) \\
&= \text{mod } (X_{i+1},\ k) \quad \text{car } k \mid n \\
&= \text{mod } (\text{mod } (aX_i + c, m),\ k) \\
&= \text{mod } (aX_i + c,\ k) \quad \text{car } k \mid m \\
&= \text{mod } (\text{mod } (aX_i + c, n),\ k) \quad \text{car } k \mid n \\
&= \text{mod } (a\ Y_i + c,\ k) \\
Y'_{i+1} &= \text{mod } (a\ Y'_i + c,\ k)
\end{aligned}
$$

De plus,

$$p \mid k \Rightarrow p \mid m \Rightarrow p \mid (a-1)$$

$$4 \mid k \Rightarrow 4 \mid m \Rightarrow 4 \mid (a-1)$$

$$(c,m) = 1 \Rightarrow (c,k) = 1$$

Donc le théorème de KNUTH nous permet d'affirmer que $\{Y'_i\}$ est une suite uniforme de nombres aléatoires $\subset [o, k-1]$

Cette structure de $\{Y_i\}$ modulo k permet de trouver une coloration de G :

pour tout i, définissons $f(x_{Y_i}) = \text{mod } (i,k)$. Comme pour tout j t q $o \leqslant j < n$, il existe $i \geqslant o$ t q. $Y_i = j$ tout sommet reçoit une couleur par cette procédure.

Puisque $\{Y'_i\}$ est une suite uniforme de nombres aléatoires sur l'intervalle $[0, k-1]$, nous savons que si $Y_i = Y_j$ alors $Y'_i = Y'_j$ et mod (i,k) = mod (j,k) et donc que $f(x_{Y_i}) = f(x_{Y_j})$. Ceci signifie que f est bien définie.

Finalement, on peut facilement vérifier que x_{Y_i} , $x_{Y_{i+1}}$, ..., $x_{Y_{i+h-1}}$ sont tous colorés différemment si $h < k$ pour tout $i \geq o$ .Cela veut dire qu'une arête joint seulement deux noeuds colorés différemment et que f est une coloration exacte de G.

Donc $\chi(G) = k$.

ANNEXE 2.1

TABLEAUX RESUMANT LES PERFORMANCES DES METHODES NON-EXACTES

	CLASSE I		CLASSE V		CLASSE IX		CLASSE XIII	
	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps
LF	4.40	0.02	6.80	0.07	8.20	0.13	9.80	0.21
LFI	4.00	0.06	6.00	0.58	6.80	2.34	8.00	8.00
SL	4.00	0.03	7.00	0.09	8.20	0.17	8.40	0.28
SLI	4.00	0.03	5.80	0.61	6.60	2.99	6.20	5.45
DFS	4.60	0.02	7.80	0.07	9.80	0.12	10.80	0.20
DFSI	4.00	0.06	6.60	1.33	7.60	7.39	8.20	15.88
BFS	4.20	0.01	7.80	0.05	9.40	0.10	10.20	0.16
BFSI	4.00	0.03	6.60	1.10	8.00	5.07	8.60	9.83
DGEN	4.00	0.03	7.00	0.09	7.60	0.16	9.20	0.26
DGENI	4.00	0.04	5.80	0.58	7.00	2.90	6.40	5.38
RND	4.60	0.02	8.20	0.08	9.20	0.13	10.20	0.20
RNDI	4.00	0.10	7.00	2.10	7.40	6.10	8.20	17.68
LFSC	4.00	0.00	6.80	0.03	8.20	0.07	9.80	0.12
MIS	4.00	2.00	7.60	88.37	7.20	2445.76	-	-
AMIS	4.40	0.01	8.20	0.08	9.20	0.17	9.40	0.28
SAT	4.00	0.01	6.00	0.10	6.80	0.19	7.20	0.32
DSI	4.00	0.03	5.40	0.21	6.00	0.93	5.40	0.69
AA	4.00	0.02	6.60	0.10	7.80	0.22	8.60	0.35
RLF	4.00	0.02	5.80	0.08	6.40	0.17	7.00	0.29
ENUMPA	4.00	0.24	5.20	483.28	-	-	-	-

ANNEXE 2.2.

	CLASSE III		CLASSE VII		CLASSE XI		CLASSE XV	
	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps
LI	5.00	0.03	12.20	0.13	20.80	0.26	28.60	0.43
LFI	5.00	0.06	10.40	1.06	18.20	13.12	27.20	36.32
SL	5.00	0.03	11.80	0.15	20.80	0.32	28.80	0.56
SLI	5.00	0.05	10.40	1.20	17.60	10.61	27.00	33.40
DFS	7.00	0.03	19.40	0.15	24.20	0.31	33.00	0.52
DFSI	5.80	0.17	14.80	5.89	20.20	23.89	28.80	64.36
BFS	7.60	0.02	16.80	0.12	23.20	0.26	31.80	0.44
BFSI	5.20	0.13	12.60	3.01	19.20	19.49	28.80	49.32
DGEN	5.00	0.03	12.00	0.15	20.00	0.33	28.60	0.59
DGENI	5.00	0.06	10.60	1.35	17.80	11.47	26.20	32.46
RND	6.60	0.03	16.80	0.13	23.40	0.28	32.80	0.43
RNDI	5.00	0.14	12.80	5.10	18.80	21.96	28.40	65.99
LFSC	5.00	0.01	12.00	0.05	20.60	0.10	28.00	0.19
MIS	5.00	0.46	11.40	11.80	19.80	71.30	29.40	322.19
AMIS	7.00	0.03	15.80	0.22	22.00	0.56	33.00	1.20
SAT	5.00	0.02	10.40	0.13	19.20	0.30	27.60	0.51
DSI	5.00	0.05	10.20	0.51	17.80	8.58	26.40	18.71
AA	5.00	0.01	13.00	0.11	19.80	0.27	28.00	0.46
RLF	5.00	0.03	10.80	0.20	17.80	0.54	26.80	1.29
ENIMPA	5.00	1.44	-	-	-	-	-	-

ANNEXE 2.3.

	CLASSE IV		CLASSE VIII		CLASSE XII		CLASSE XVI	
	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps	Nombre couleurs	Temps
LL	10.00	0.03	25.00	0.16	28.80	0.32	50.40	0.59
LLI	10.00	0.32	25.00	6.55	26.40	16.44	50.20	71.88
SL	10.00	0.04	25.00	0.19	28.20	0.40	51.20	0.72
SLI	10.00	0.32	25.00	6.12	26.60	15.76	50.00	73.71
DLS	11.80	0.04	30.60	0.20	36.20	0.41	58.80	0.73
DFSI	10.60	0.68	27.20	14.76	29.60	35.24	53.00	144.85
BFS	11.80	0.03	30.20	0.15	35.60	0.32	57.40	0.57
BFSI	10.60	0.63	26.60	12.48	29.80	31.24	52.60	127.89
DGEN	10.00	0.04	25.00	0.19	28.60	0.42	50.40	0.77
DGENI	10.00	0.32	25.00	6.52	26.80	17.53	50.00	71.50
RND	12.00	0.04	29.80	0.16	34.60	0.33	57.00	0.57
RNDI	10.60	0.67	26.80	14.28	29.80	36.72	53.20	146.74
LFSC	10.00	0.01	25.00	0.06	29.40	0.12	50.60	0.24
MIS	10.80	0.48	27.80	8.51	29.20	29.79	56.40	69.45
AMIS	12.00	0.05	31.20	0.43	35.20	1.00	62.00	2.82
SAT	10.00	0.03	25.00	0.17	26.80	0.36	50.00	0.67
DSI	10.00	0.31	25.00	6.42	25.80	10.64	50.00	69.45
AA	10.00	0.05	25.00	0.13	28.00	0.26	50.20	0.50
RLF	10.00	0.02	25.00	0.44	26.80	0.93	50.20	2.82
ENZIPA	-	-	-	-	-	-	-	-

REFERENCES

[1] D.S. Johnson; "Worst case behaviour of graph coloring algorithms", Proceed. of the 5th Southeast conference on combinatories, graph theory and computing.

[2] Leighton;"A graph coloring algorithm for large scheduling problems";Journal of Research of the national bureau of standards, vol. 84 nº 6, 1979, p. 489-506.

[3] Knuth D.E., "The art of computer programming"; Addison - Wesley, Reading, MA, 1969.

[4] Matula, Marble, Isaacson; "Graph coloring algorithms"; Graph Theory and Computing.

[5] Welsh, Powell, "An Upper bound for the chromatic number of a graph and its applications to timetable problems"; the computer journal, 1967, vol.10, p.85-86.

[6] M.R. Williams; "The coloring of very Large graphs"; Combinational structures and their applications, p. 477-478.

[7] D.S. Jonhson; "Approximation Algorithms for Combinational Problems";Journal of Computer and System Sciences, 9,p. 256-278, 1974.

[8] D. Brelaz; "New methods to color the vertices of a graph"; Communications of the ACM, 1979, vol. 22, nº4, p.251-256.

[9] Tehrani A.; "Un algorithme de coloration"; Cahiers du Centre d'Etudes et de Recherche Opérationnelle, vol. 117, 1975, p. 395-398.

[10] O. Henkes; "Etablissement d'horaires de cours par coloration de graphes à matrice de temps"; Revue Belge de Statistique, d'Informatique et de Recherche Opérationnelles, Vol. 24, nº 2.

[11] N. Christofides "An Algorithm for the chromatic number of a graph"; the computer journal, 1970, vol.14, nº 1, p. 38-39.

[12] B. Carré; "Independent Sets, dominating sets and colorations"; Graphs and Networks,p. 175-197.

[13] C. Wang "An algorithm for the chromatic number of a graph"; Journal of the ACM, vol. 21, n°3, 1974, p. 385-391.

[14] J. Pfeemöler; "A correction to Brelaz's modification of Brown's coloring algorithm"; Communications of the ACM, vol. 26, n° 8, 1983.

[15] Aho, Hopcroft, Ullman; "The design and Analysis of Computer Algorithms"; Addison-Wesley, Reading, MA, 1974, p. 364-404.

- - - - -

BIBLIOGRAPHIE

J. RANDALL BROWN; "Chromatic scheduling and the chromatic number problem"; Management Science, 1972, p. 456-463.

R.L. BROOKS; "On colouring the Nodes of a Network"; Proceed. Cambridge Phil. Soc., vol. 37, 1941, p. 194-197.

WILF, SZEKERES; "An Inequality for the Chromatic Number of a Graph"; Journal of the Combinatorial Theory 4, 1968, p. 1-3.

J.A. BONDY; "Bounds for the Chromatic Number of a Graph"; Journal of the Combinatorial Theory 7, 1969, p. 96-98.

D.W. MATULA; "A Min-Max theorem for graphs with applications to graph coloring"; Chronicle SIAM Revue 1968, vol. 10, p. 481-482.

P.A. CATLIN; "A bound on the chromatic number of a graph"; Discrete Mathematics 22, 1978, p. 81-82.

I. TOMESCU; "On the chromatic number of almost all graphs"; Bulletin Mathématique de la Société des Sciences Math. et Phys. de la Roumanie, 1981, vol. 25, n° 3, p. 321-323.

DE WERRA; "How to color a Graph"; Combinatorial Programming : Methods and Applications, NLD, 1975, p. 305-325.

HANSEN, DELATTRE; "Complete-link cluster analysis by graph coloring"; J.A.S.A., 1978, vol. 73, n° 362, p. 397-403.

GEOFFREY GRIMMET; "Random Graph Theorems"; Proceed. of the 7-th Prague Conference on information theory and related topics, 1974, p. 203-209.

NIJENHUIS, WILF; "Chromatic Polynom of a graph"; Combinatorial Algorithms, Academic Press, p. 118-133.

O.YU.PERSHIN; "An algorithm for determining the minimum colouring of a finite graph"; Eng. Cybernetics, 1973, vol. 11, n° 6, p. 980-985.

Mc DIARMID; "Determinig the chromatic number of a graph"; SIAM J. Computing, vol. 8, n° 1, 1979, p. 1-14.

Mc DIARMID, GRIMMET; "On colouring random graphs"; Math. Proceed. Cambridge soc. 1975, vol. 77, p. 313-324.

CHIBA, NISHZEKI, SAITO; "An approximation Algorithm for the max. indep. set problem of planar graphs"; SIAM J. Computing, vol. 11, n° 4, 1982, p. 663-675.

A.J. HOFFMANN; " On Eigenvalues and Graph Colourings"; Graph Theory and Computing, p. 79-91.

H.S. WILF; "The Eigenvalues of a graph and its Chromatic Number"; J. London Math. Society, 42, 1967, p. 330-332.

O. HENKES ; "Etablissement d'horaires de cours par coloration de graphes"; Mémoire présenté pour l'obtention du titre de Licencié et Maître en Informatique, F.N.D.P. Namur, 1982.

D.W. MATULA; "Bounded Color Functions on Graphs"; Networks, vol. 2, 1972 - p. 29-44.

ROSCHKE, FURTADO; "An algorithm for obtaining the chromatic number of a graph"; Information processing letters, 2, 1973, p. 34-38.

CORNEIL, GRAHAM; "An algorithm for determining the chromatic number of a graph"; SIAM J. Computing, vol. 2, n° 41, 1973. p. 311-318.

J. MITCHEM; "On various algorithms for estimating the chromatic number of a graph"; Computer journal, vol. 19, n° 2, p. 182-193.

SAKATI, NAKASHIMA; "Algorithms for defining in the lump boths bounds of the chromatic number of a graph"; Computer journal, vol. 19, n° 4, p. 329-331.

M.R. WILLIAMS; "A bound for the Chromatic Number of a graph"; Thesis Glasgow, p.125-131.

- - - - - - - - - -